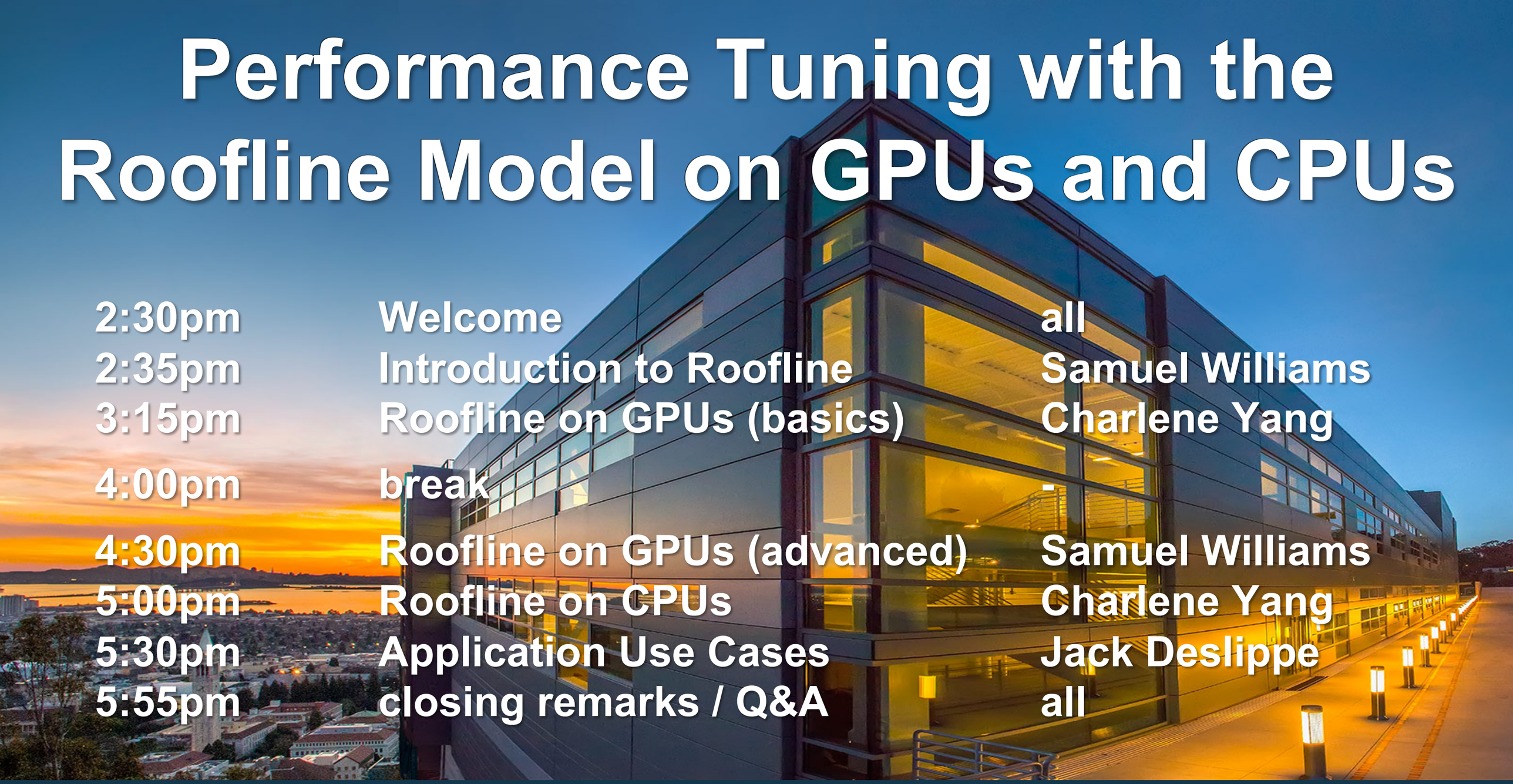
Performance Tuning with the Roofline Model on GPUs and CPUs



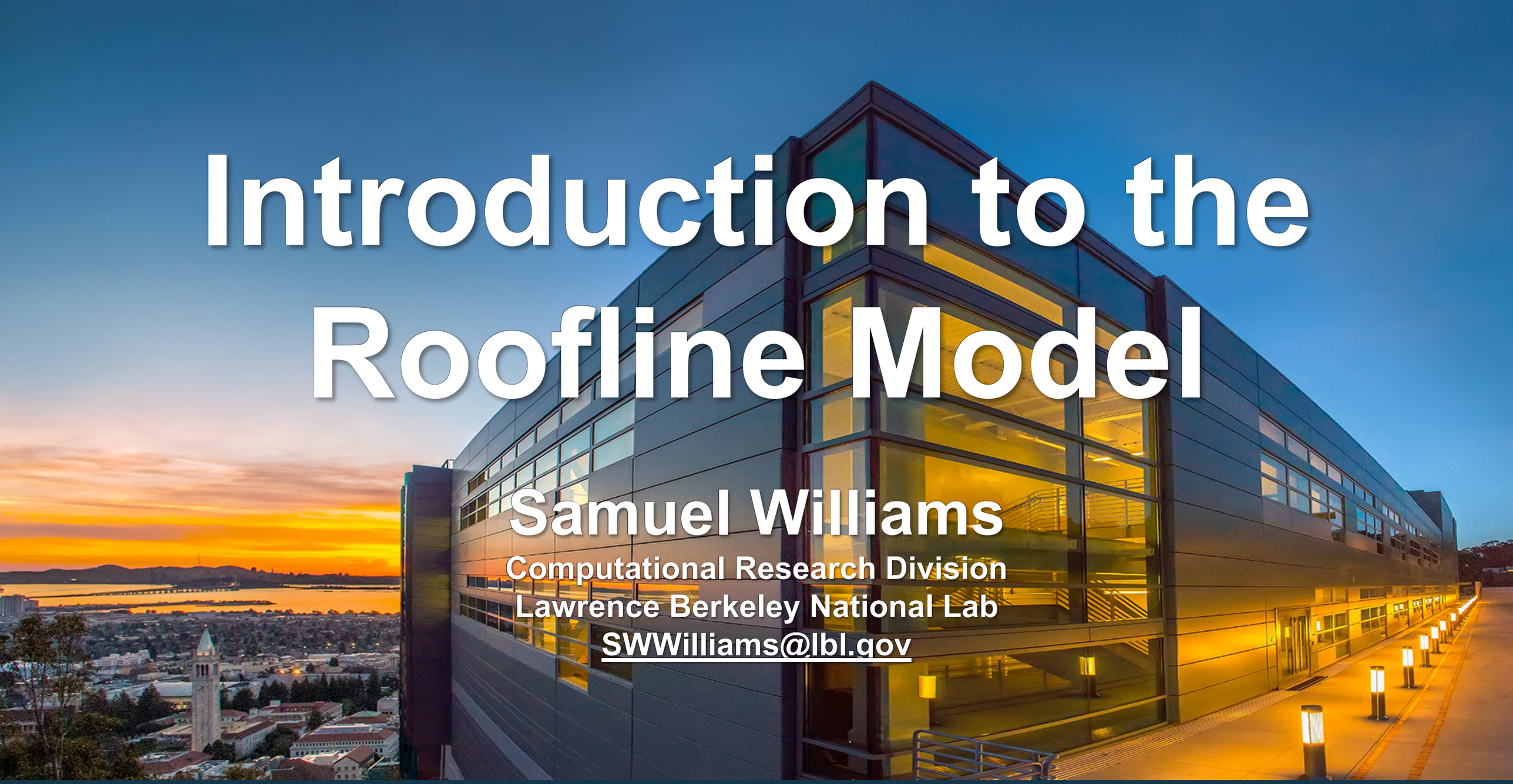
2:30pm	Welcome	all
2:35pm	Introduction to Roofline	Samuel Williams
3:15pm	Roofline on GPUs (basics)	Charlene Yang
4:00pm	break	-
4:30pm	Roofline on GPUs (advanced)	Samuel Williams
5:00pm	Roofline on CPUs	Charlene Yang
5:30pm	Application Use Cases	Jack Deslippe
5:55pm	closing remarks / Q&A	all

Introductions

Samuel Williams
Computational Research Division
Lawrence Berkeley National Lab
SWWilliams@lbl.gov

Charlene Yang
NERSC
Lawrence Berkeley National Lab
CJYang@lbl.gov

Jack Deslippe
NERSC
Lawrence Berkeley National Lab
JRDeslippe@lbl.gov



Introduction to the Roofline Model

Samuel Williams

Computational Research Division
Lawrence Berkeley National Lab
SWWilliams@lbl.gov

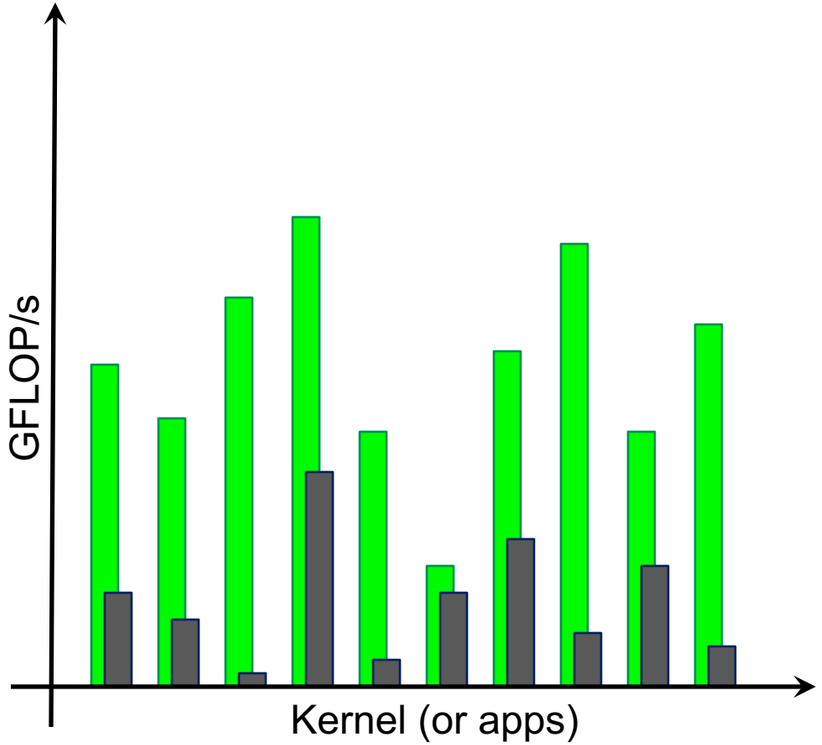
Acknowledgements

- This material is based upon work supported by the Advanced Scientific Computing Research Program in the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231.
- This material is based upon work supported by the DOE RAPIDS SciDAC Institute.
- This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.
- This research used resources of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

What is “Good” Performance?

- Imagine running a mix of benchmarks on a new system (e.g. GPU/CPU)…
  - GFLOP/s alone may not be particularly insightful
  - speedup relative to the previous system may seem random

➢ **We need a quantitative model that defines Good Performance**



What is “Good” Performance?

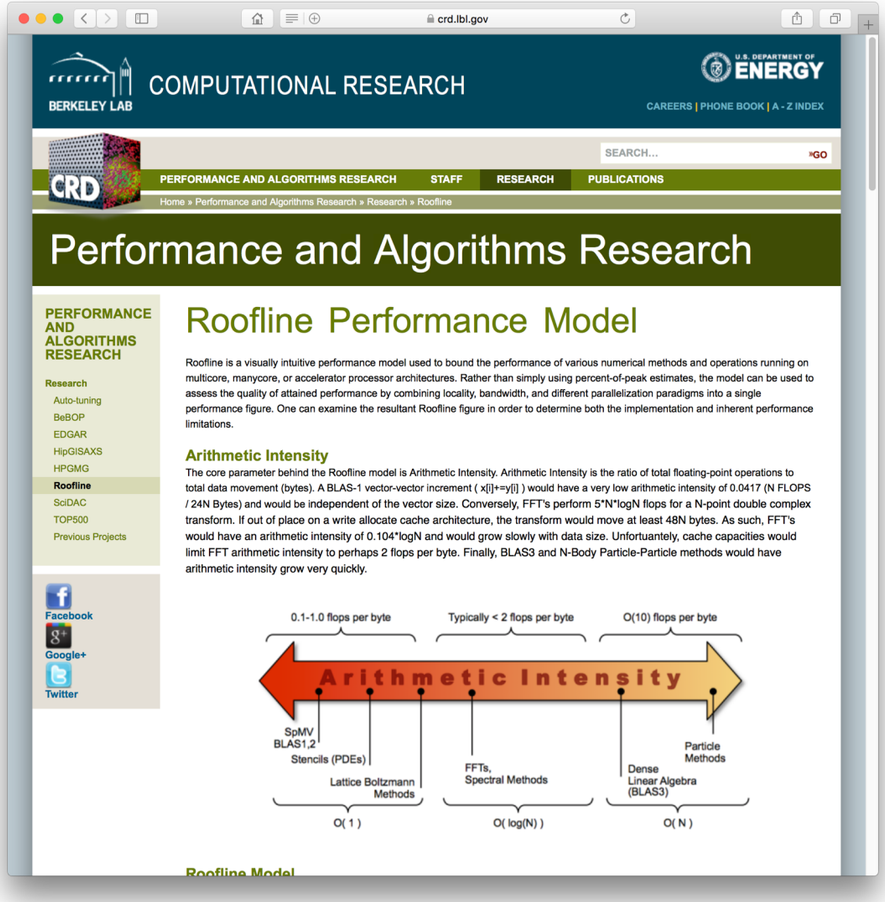
- Good Performance is tied to “Efficient” execution
- Two fundamental requirements …

1. Must operate the CPU/GPU in the throughput-limited regime
   not sensitive to Amdahl effects, D2H/H2D transfers, launch overheads, etc…

2. Must attain high utilization of the CPU/GPU’s **compute** and/or **bandwidth** capabilities

Roofline Model

- **Roofline Model** is a throughput-oriented performance model
- applies to x86, ARM, POWER CPUs, GPUs, Google TPUs[1], FPGAs, etc…
- Helps quantify **Good Performance**



https://crd.lbl.gov/departments/computer-science/PAR/research/roofline

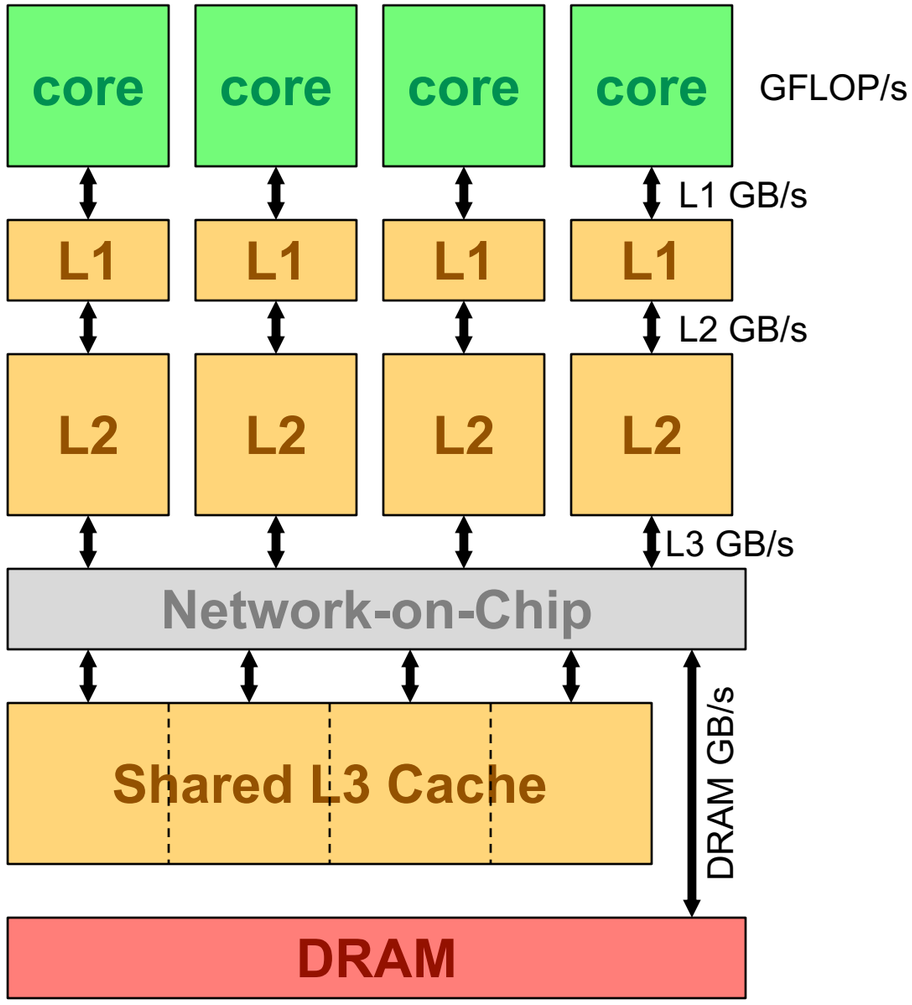
[1]Jouppi et al, "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA, 2017.

Reduced Model

- Modern architectures can be complex
- Don’t model / simulate full architecture
- Make assumptions on performance and usage…

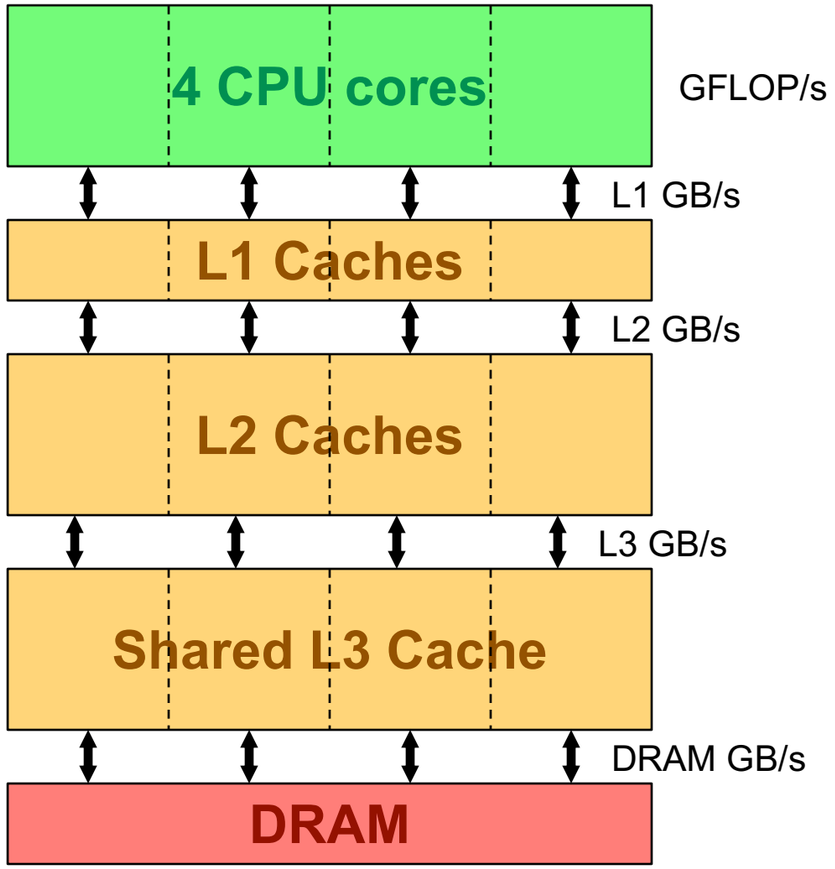
Reduced Model

- Modern architectures can be complex
- Don’t model / simulate full architecture
- Make assumptions on performance and usage…
  - Peak GFLOP/s on data in L1



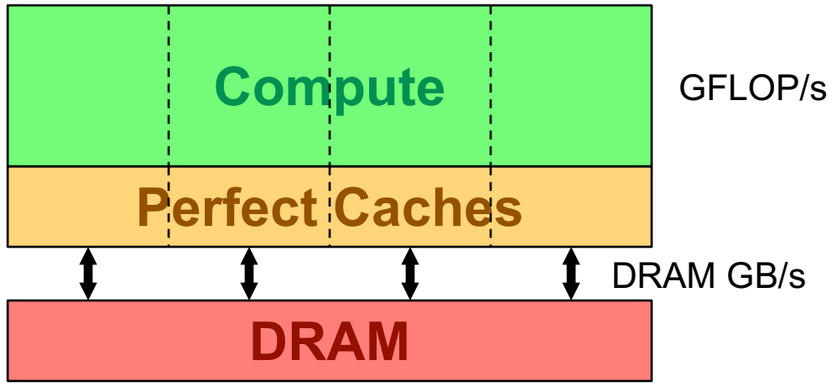
Reduced Model

- Modern architectures can be complex
- Don’t model / simulate full architecture
- Make assumptions on performance and usage…
  - Peak GFLOP/s on data in L1
  - Load-balanced SPMD code



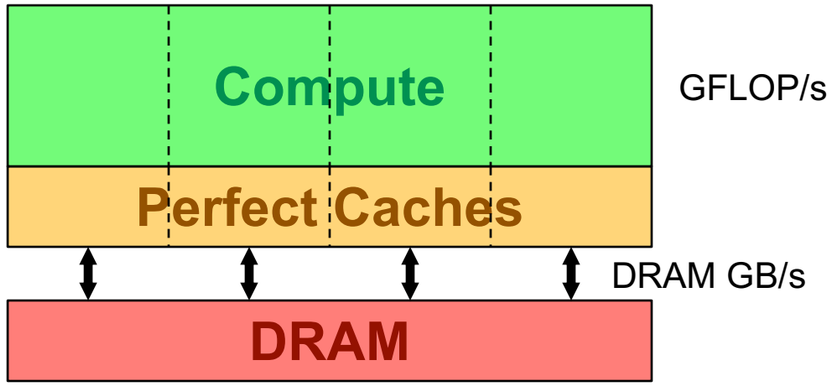
Reduced Model

- Modern architectures can be complex
- Don’t model / simulate full architecture
- Make assumptions on performance and usage…
  - Peak GFLOP/s on data in L1
  - Load-balanced SPMD code
  - Sufficient cache bandwidth/capacity



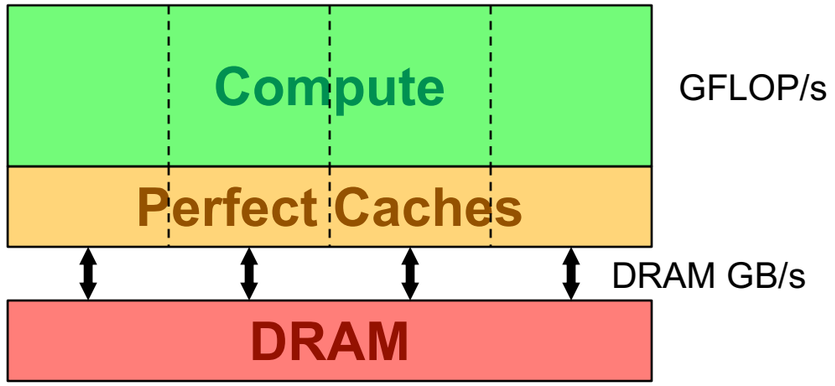
Reduced Model

- Modern architectures can be complex
- Don’t model / simulate full architecture
- Make assumptions on performance and usage…
  - Peak GFLOP/s on data in L1
  - Load-balanced SPMD code
  - Sufficient cache bandwidth/capacity
  - ➢ **Basis for DRAM Roofline Model**



(DRAM) Roofline

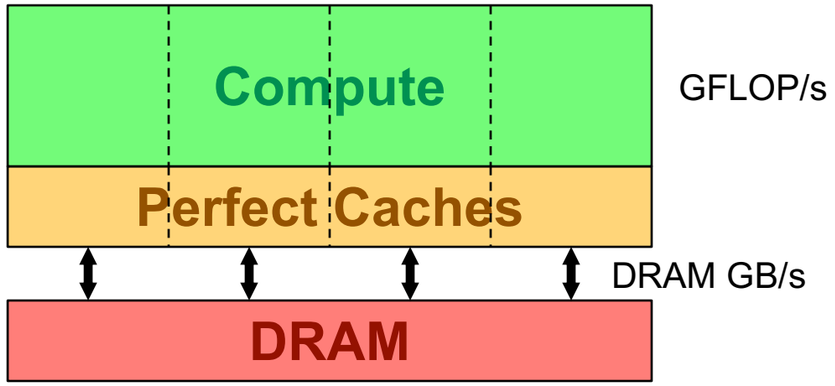
- Any given loop nest will perform:
  - Computation (e.g. FLOPs)
  - Communication (e.g. moving data to/from DRAM)
- With perfect overlap of communication and computation...
  - Run time is determined by whichever is greater



$$\text{Time} = \max \begin{cases} \#\text{FLOPs} / \text{Peak GFLOP/s} \\ \#\text{Bytes} / \text{Peak GB/s} \end{cases}$$

(DRAM) Roofline

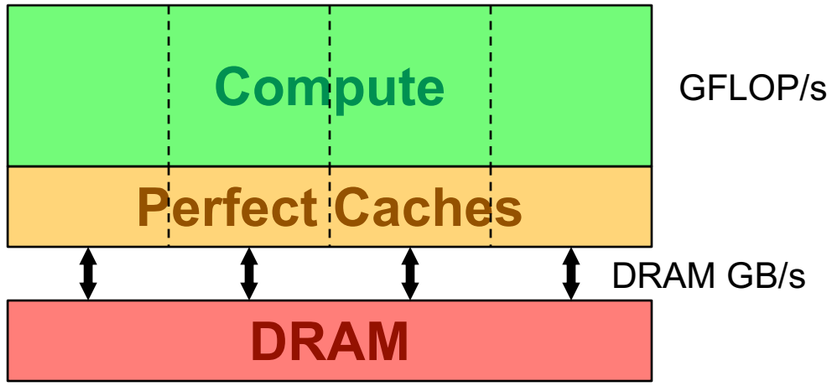
- Any given loop nest will perform:
  - Computation (e.g. FLOPs)
  - Communication (e.g. moving data to/from DRAM)
- With perfect overlap of communication and computation...
  - Run time is determined by whichever is greater



$$\frac{\text{Time}}{\#\text{FLOPs}} = \max \begin{cases} 1 \,/\, \text{Peak GFLOP/s} \\ \#\text{Bytes} \,/\, \#\text{FLOPs} \,/\, \text{Peak GB/s} \end{cases}$$

(DRAM) Roofline

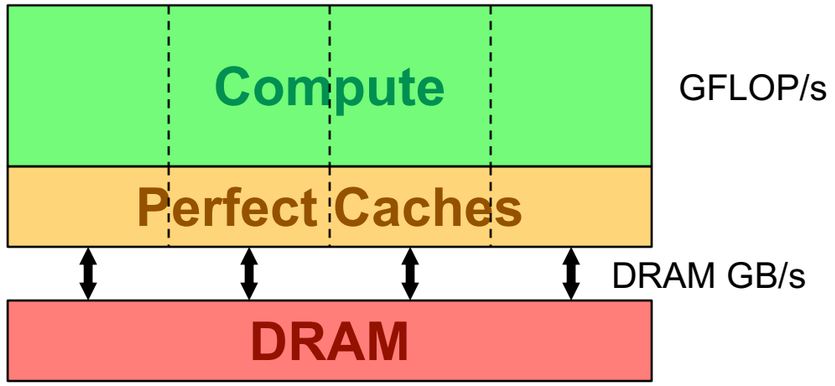
- Any given loop nest will perform:
  - Computation (e.g. FLOPs)
  - Communication (e.g. moving data to/from DRAM)
- With perfect overlap of communication and computation...
  - Run time is determined by whichever is greater



$$\frac{\#\text{FLOPs}}{\text{Time}} = \min \begin{cases} \text{Peak GFLOP/s} \\ (\#\text{FLOPs} / \#\text{Bytes}) * \text{Peak GB/s} \end{cases}$$

(DRAM) Roofline

- Any given loop nest will perform:
  - Computation (e.g. FLOPs)
  - Communication (e.g. moving data to/from DRAM)
- With perfect overlap of communication and computation...
  - Run time is determined by whichever is greater



$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI} * \text{Peak GB/s} \end{cases}$$

AI (Arithmetic Intensity) = FLOPs / Bytes (as presented to DRAM)

Arithmetic Intensity

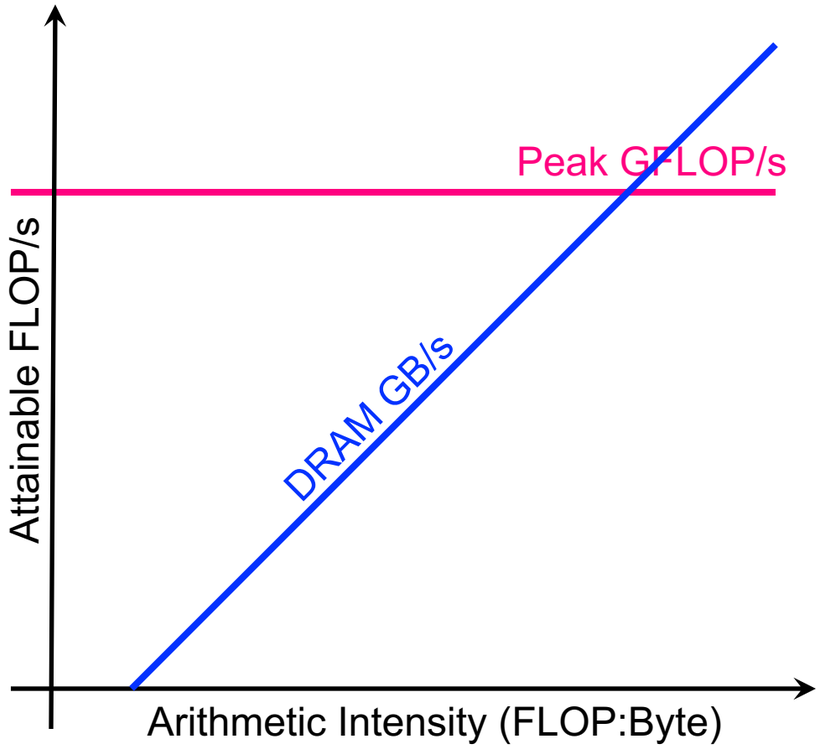
- Measure of data locality (data reuse)
- Ratio of **Total Flops** performed to **Total Bytes** moved
- For the DRAM Roofline…
  - Total Bytes to/from DRAM
  - Includes all cache and prefetcher effects
  - Can be very different from total loads/stores (bytes requested)
  - Equal to ratio of sustained GFLOP/s to sustained GB/s (time cancels)

(DRAM) Roofline Model

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI} * \text{Peak GB/s} \end{cases}$$

AI (Arithmetic Intensity) = FLOPs / Bytes (moved to/from DRAM)

- Plot Roofline bound using Arithmetic Intensity as the x-axis
- **Log-log scale** makes it easy to doodle, extrapolate performance along Moore's Law, etc…

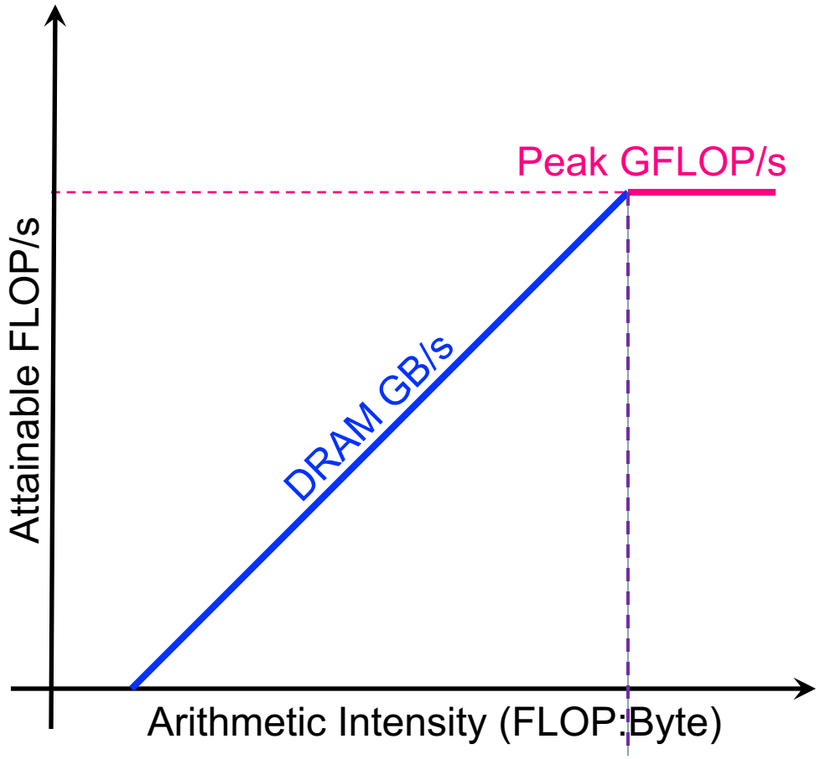


(DRAM) Roofline Model

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI} * \text{Peak GB/s} \end{cases}$$

AI (Arithmetic Intensity) = FLOPs / Bytes (moved to/from DRAM)

- Plot Roofline bound using Arithmetic Intensity as the x-axis
- **Log-log scale** makes it easy to doodle, extrapolate performance along Moore's Law, etc…



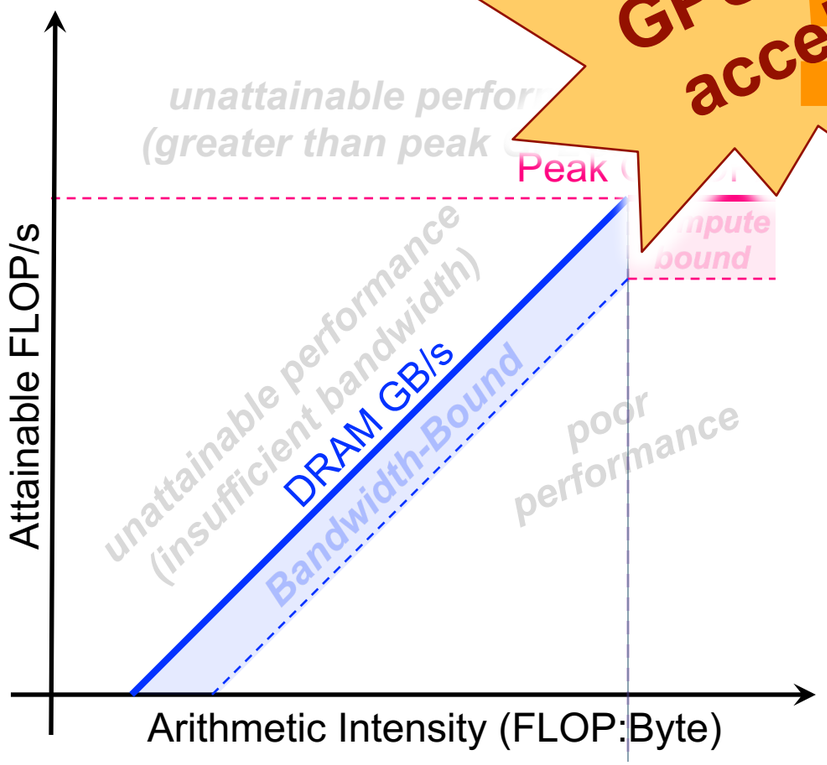
Transition @ AI == Peak GFLOP/s / Peak GB/s == 'Machine Balance'

(DRAM) Roofline Model

GFLOP/s = min { Peak GFLOP/s ; AI * Peak GB/s }

AI (Arithmetic Intensity) = FLOPs / Bytes (moved to/from DRAM)

- Roofline tessellates this 2D view of performance into 5 regions…



Applies to GPUs or other accelerators!

Transition @ AI == Peak GFLOP/s / Peak GB/s == 'Machine Balance'

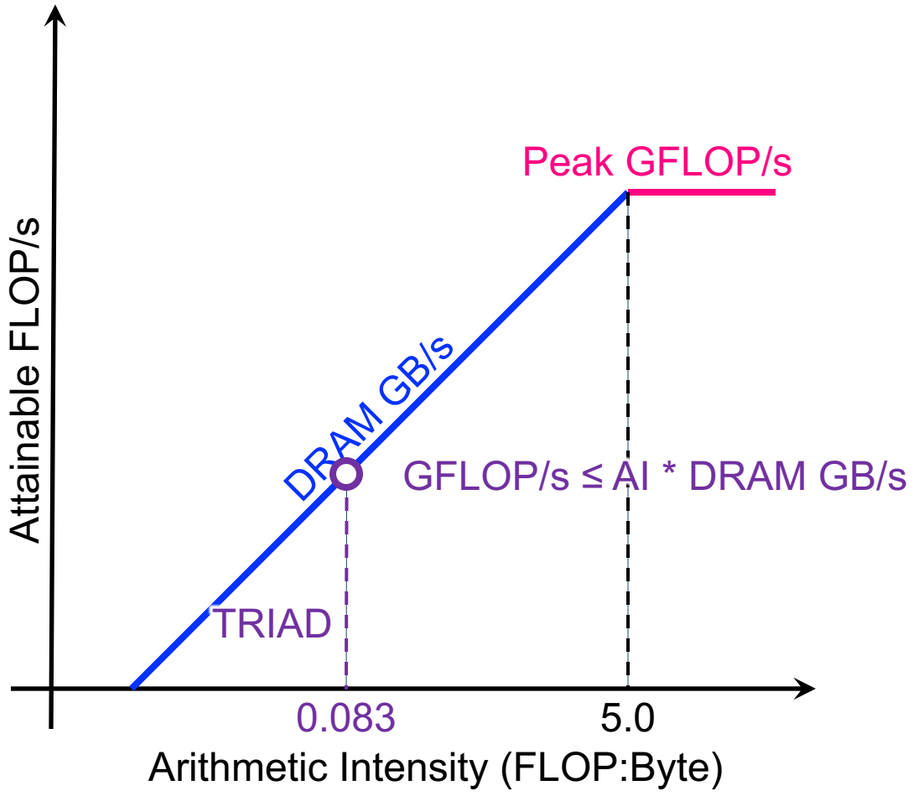
Roofline Example #1

- Typical machine balance is 5-10 FLOPs per byte…
  - 40-80 FLOPs per double to exploit compute capability
  - Artifact of technology and money
  - **Unlikely to improve**

- Consider STREAM Triad…

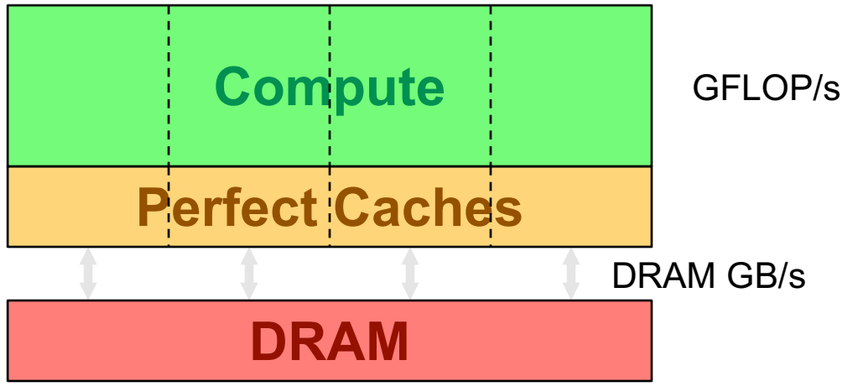
```
#pragma omp parallel for
for(i=0;i<N;i++){
  Z[i] = X[i] + alpha*Y[i];
}
```

  - 2 FLOPs per iteration
  - Transfer 24 bytes per iteration (read X[i], Y[i], write Z[i])
  - **AI = 0.083 FLOPs per byte == Memory bound**



Roofline Example #2

- Conversely, 7-point constant coefficient stencil…

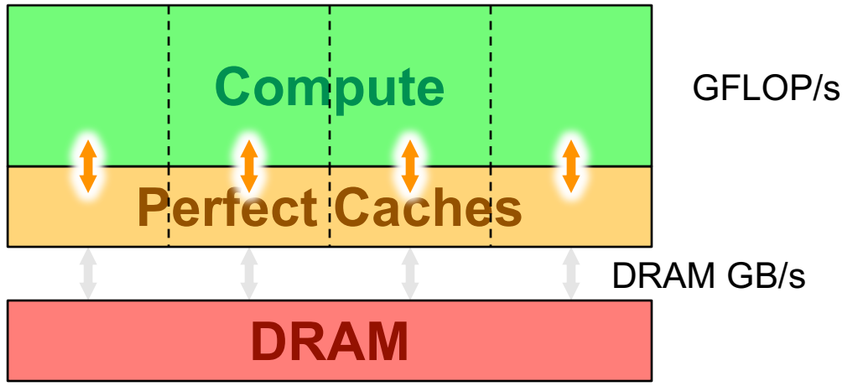



```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                 + old[k  ][j  ][i-1]
                 + old[k  ][j  ][i+1]
                 + old[k  ][j-1][i  ]
                 + old[k  ][j+1][i  ]
                 + old[k-1][j  ][i  ]
                 + old[k+1][j  ][i  ];
}}}
```

Roofline Example #2

- Conversely, 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - **AI = 7 / (8*8) = 0.11 FLOPs per byte**

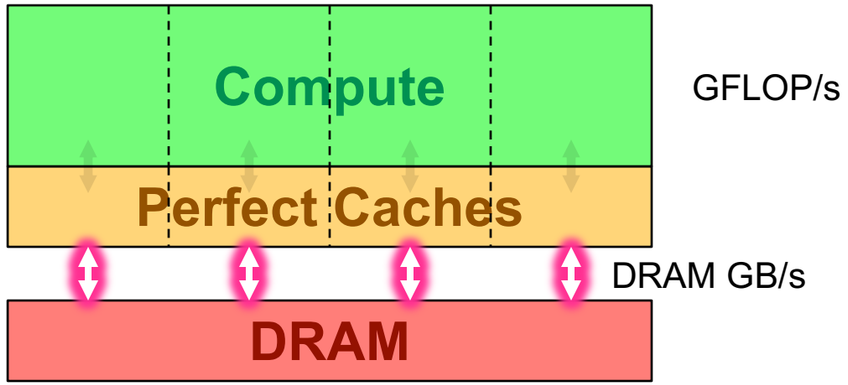
    (measured at the L1)




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
 new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                + old[k  ][j  ][i-1]
                + old[k  ][j  ][i+1]
                + old[k  ][j-1][i  ]
                + old[k  ][j+1][i  ]
                + old[k-1][j  ][i  ]
                + old[k+1][j  ][i  ]
}}}
```

Roofline Example #2

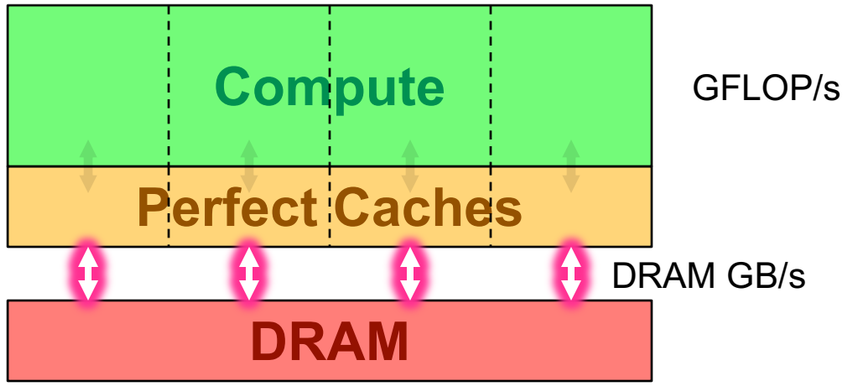
- Conversely, 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - **Ideally, cache will filter all but 1 read and 1 write per point**




```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
 new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                + old[k  ][j  ][i-1]
                + old[k  ][j  ][i+1]
                + old[k  ][j-1][i  ]
                + old[k  ][j+1][i  ]
                + old[k-1][j  ][i  ]
                + old[k+1][j  ][i  ]
}}}
```

Roofline Example #2

- Conversely, 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - Ideally, cache will filter all but 1 read and 1 write per point
  - ➢ **7 / (8+8) = 0.44 FLOPs per byte (DRAM)**



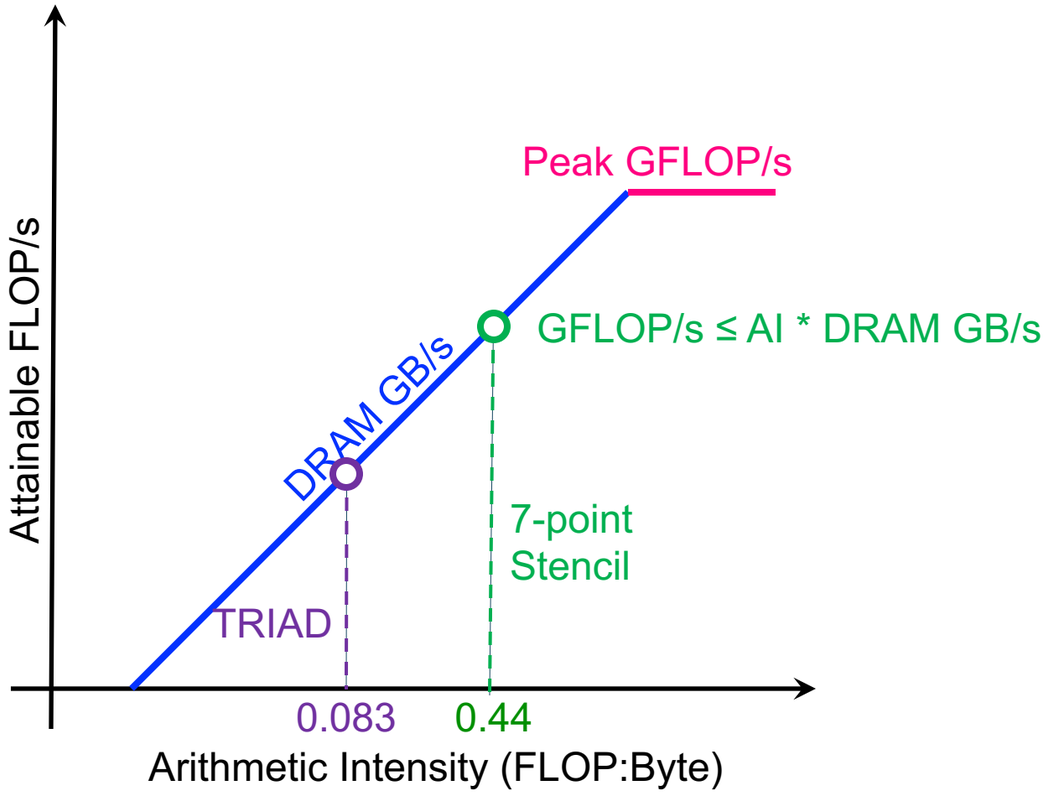

```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                 + old[k  ][j  ][i-1]
                 + old[k  ][j  ][i+1]
                 + old[k  ][j-1][i  ]
                 + old[k  ][j+1][i  ]
                 + old[k-1][j  ][i  ]
                 + old[k+1][j  ][i  ];
}}}
```

Roofline Example #2

- Conversely, 7-point constant coefficient stencil…
  - 7 FLOPs
  - 8 memory references (7 reads, 1 store) per point
  - Ideally, cache will filter all but 1 read and 1 write per point
  - **7 / (8+8) = 0.44 FLOPs per byte (DRAM)**

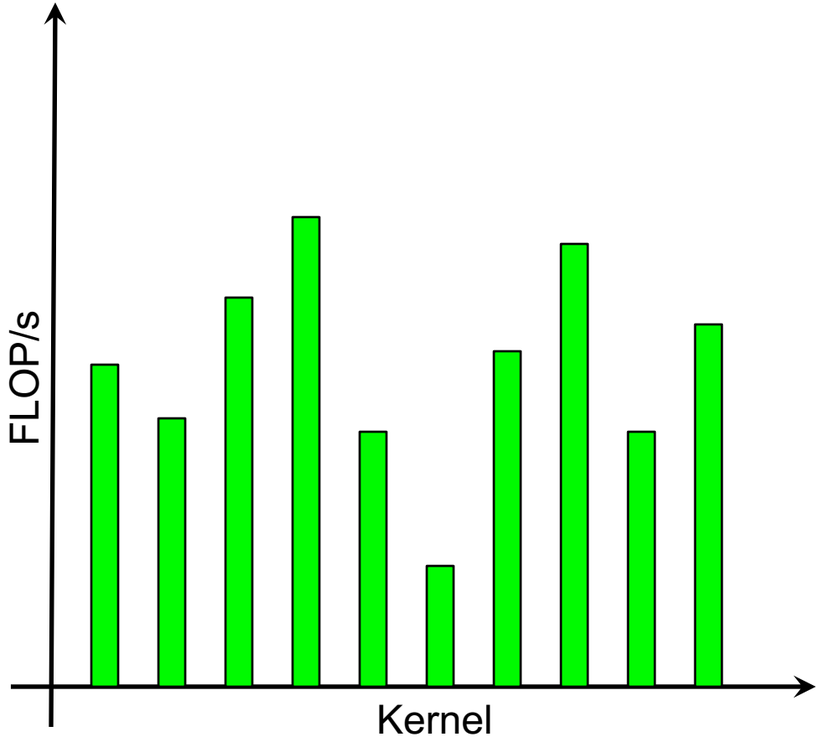
    == memory bound, but 5x the FLOP rate as TRIAD

```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
for(j=1;j<dim+1;j++){
for(i=1;i<dim+1;i++){
  new[k][j][i] = -6.0*old[k  ][j  ][i  ]
                 + old[k  ][j  ][i-1]
                 + old[k  ][j  ][i+1]
                 + old[k  ][j-1][i  ]
                 + old[k  ][j+1][i  ]
                 + old[k-1][j  ][i  ]
                 + old[k+1][j  ][i  ];
}}}
```



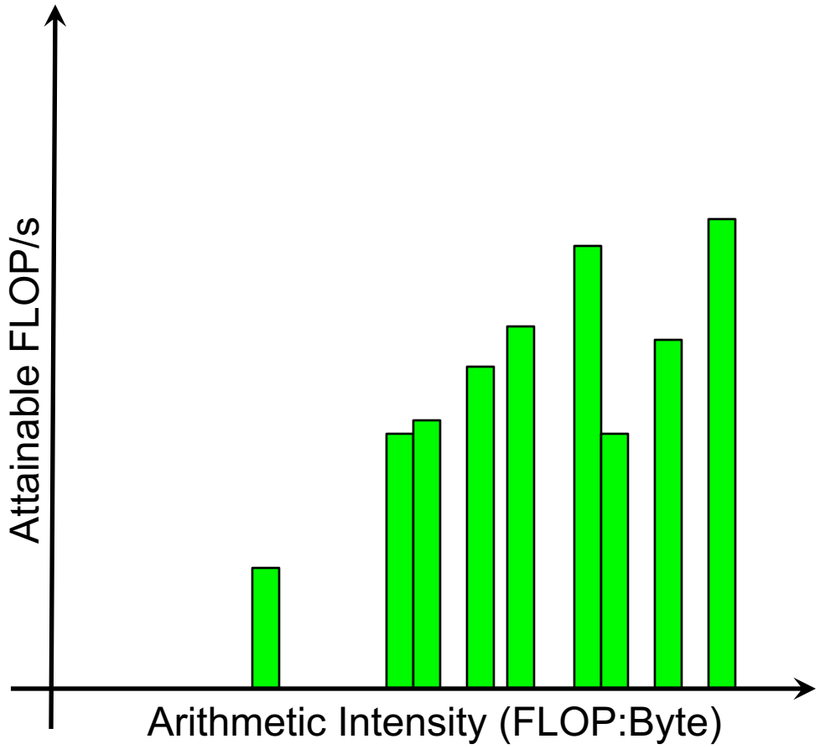
What is “Good” Performance?

- Think back to our mix of loop nests (benchmarks)…



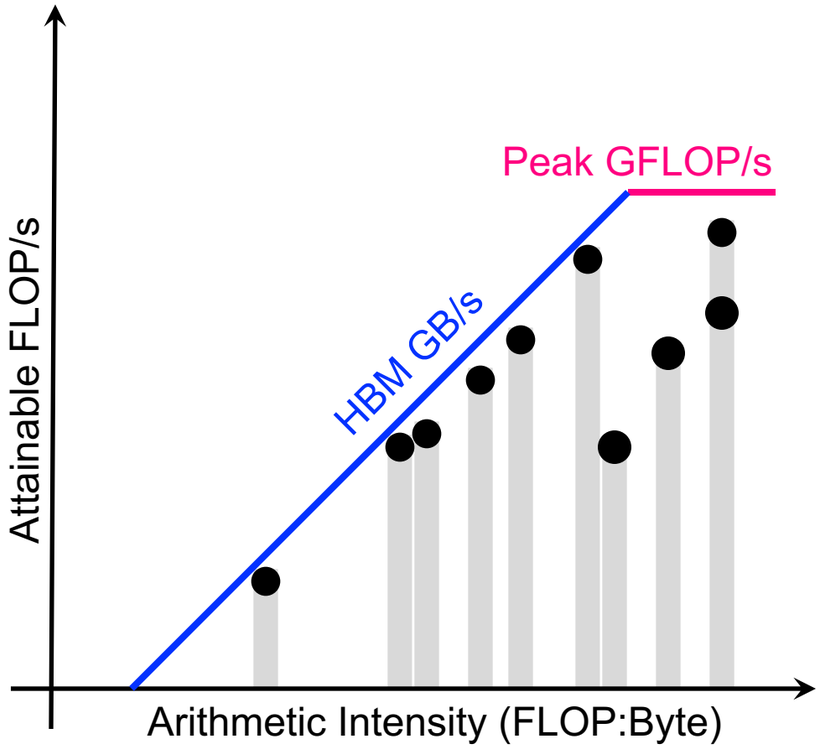
What is “Good” Performance?

- Think back to our mix of loop nests (benchmarks)
- We can sort kernels by their arithmetic intensity…



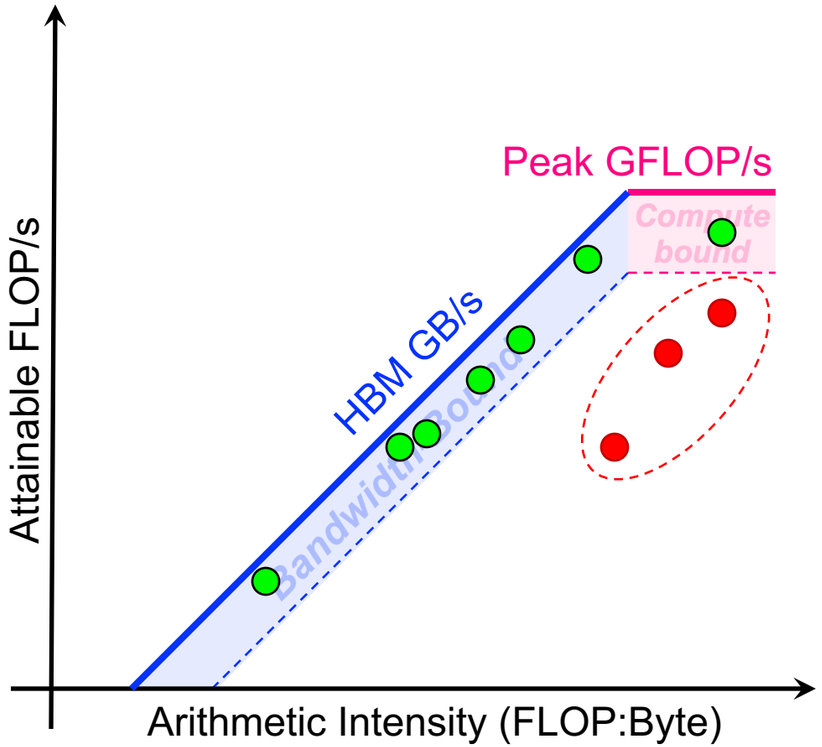
What is “Good” Performance?

- Think back to our mix of loop nests (benchmarks)
- We can sort kernels by their arithmetic intensity…
- … and compare performance relative to machine capabilities



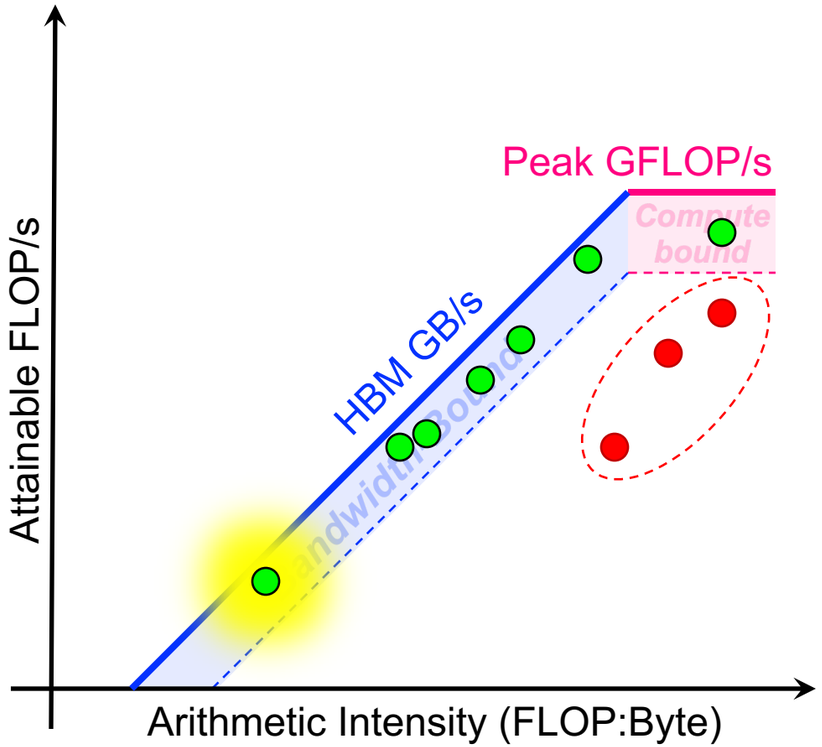
What is “Good” Performance?

- Kernels near the roofline are making **good use** of computational resources



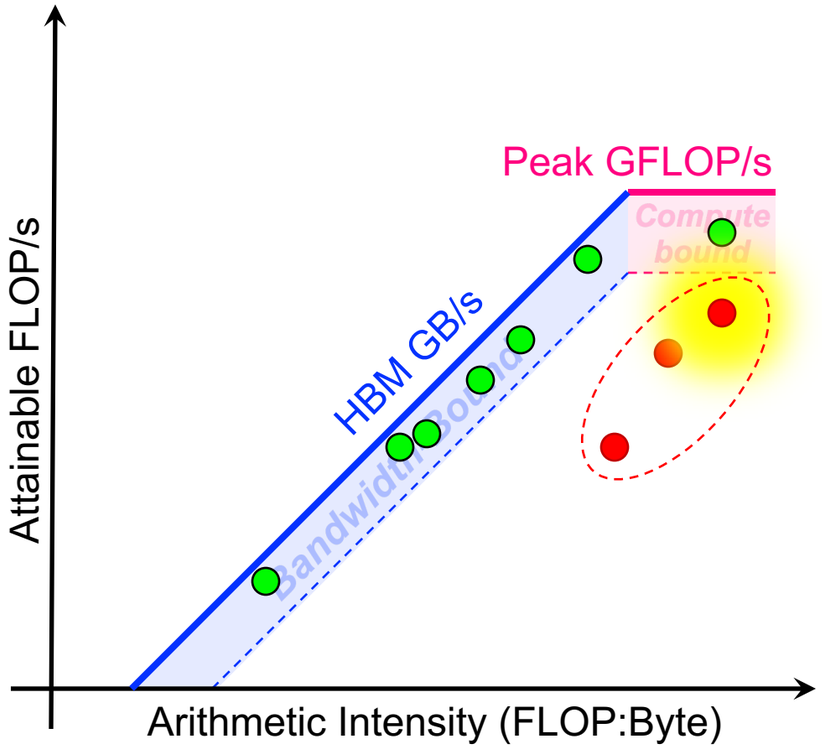
What is “Good” Performance?

- Kernels near the roofline are making **good use** of computational resources
  - ➢ kernels can have **low performance** (GFLOP/s), but make **good use** (%STREAM) of a machine



What is “Good” Performance?

- Kernels near the roofline are making **good use** of computational resources
  - kernels can have **low performance** (GFLOP/s), but make **good use** (%STREAM) of a machine
  - kernels can have **high performance** (GFLOP/s), but still make **poor use** of a machine (%peak)

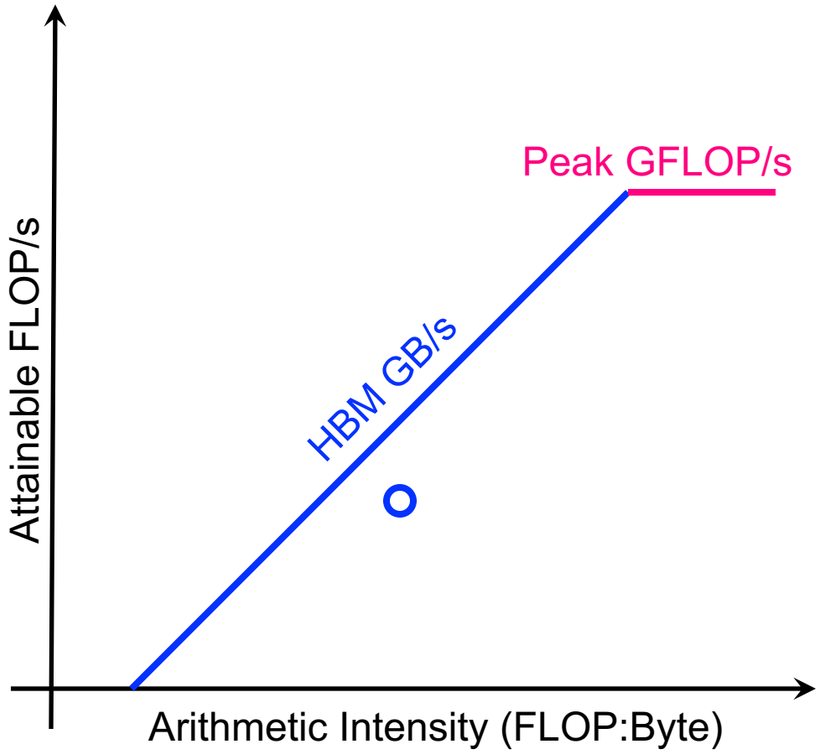



How can performance ever be below the Roofline?

How can performance be below the Roofline?

- Does one always attain either…
  - Peak DRAM Bandwidth
  - Peak FLOP/s
- Theoretical vs. Empirical
  - Use benchmarked GFLOP/s and GB/s
  - Application FLOPs can be underestimated (how many FLOPs is a divide?)
- Bottlenecks other than DRAM and FLOP/s…
  - Insufficient cache bandwidth + locality
  - Didn't use FMA / Vectors / Tensors / …
  - Too many non-FP instructions
  - Load imbalance; not SPMD

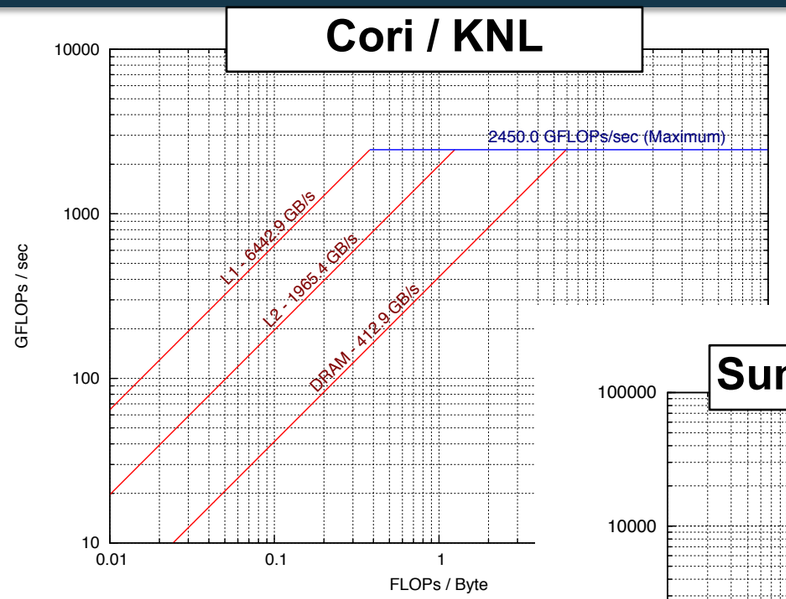
etc…

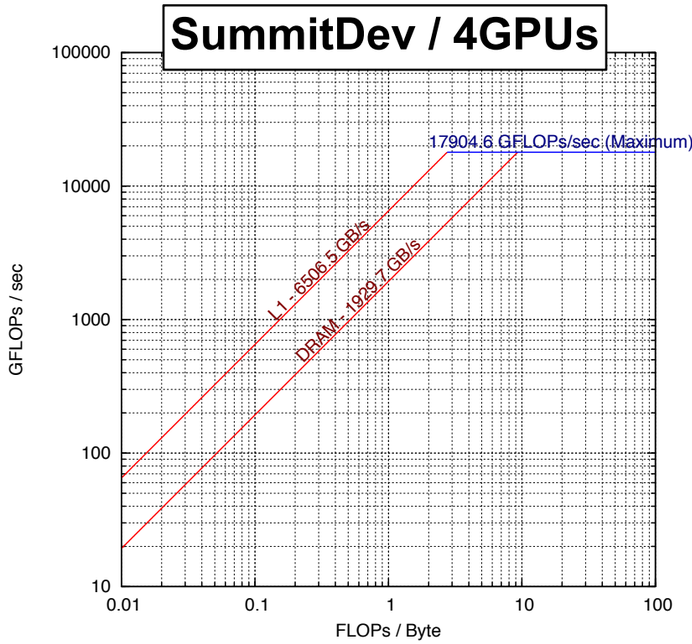



Below the Roofline?
Theoretical vs. Empirical

Machine Characterization

- Theoretical performance (specs) can be highly optimistic…
  - DRAM pin bandwidth vs. sustained
  - TurboMode / Underclocking
  - compiler failing on high-AI loops.
- Need empirical performance data
- LBL developed the Empirical Roofline Toolkit (ERT)…
  - Characterize CPU/GPU systems
  - Peak Flop rates
  - Bandwidths for each level of memory
  - **MPI+OpenMP/CUDA == multiple GPUs**

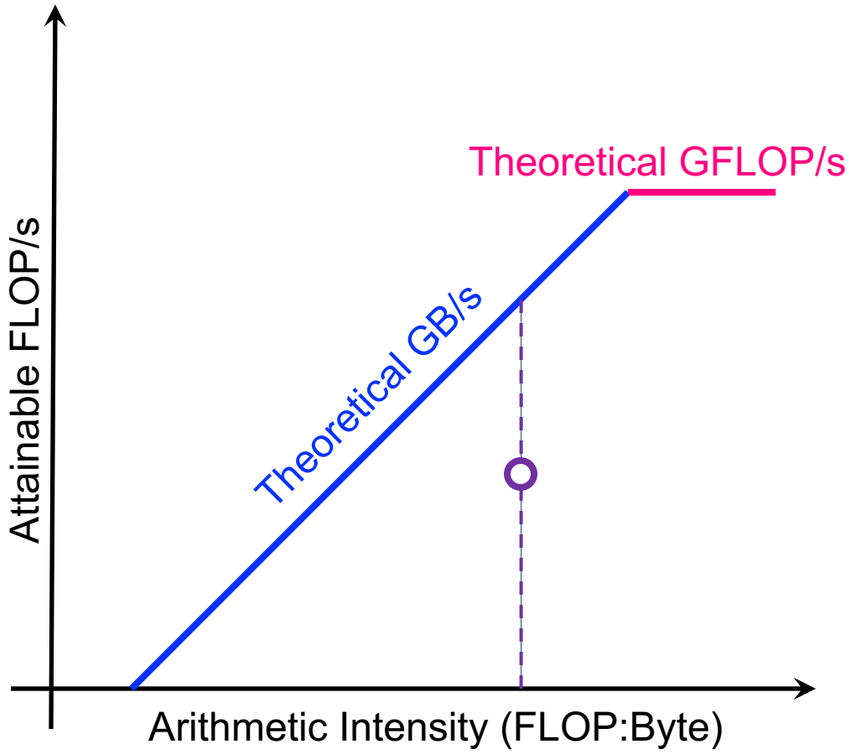




https://bitbucket.org/berkeleylab/cs-roofline-toolkit
https://github.com/cyanguwa/nersc-roofline
https://crd.lbl.gov/departments/computer-science/PAR/research/roofline

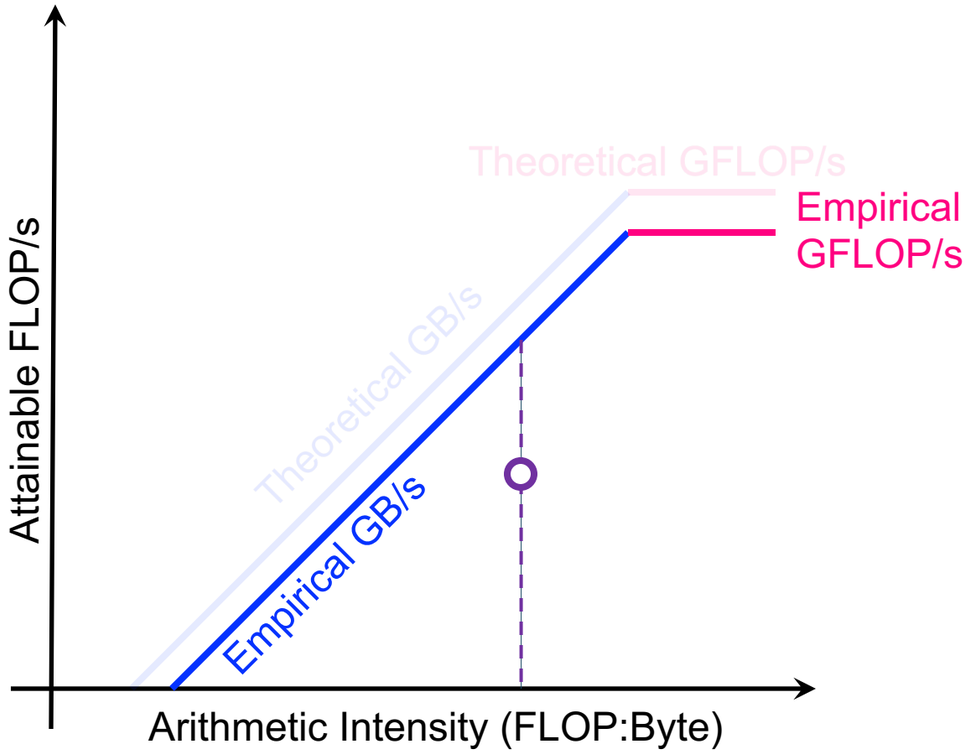
Theoretical vs. Empirical

- Theoretical Roofline:
  - Pin bandwidth
  - FPUs * GHz
  - 1 C++ FLOP = 1 ISA FLOP
  - Data movement = Compulsory Misses



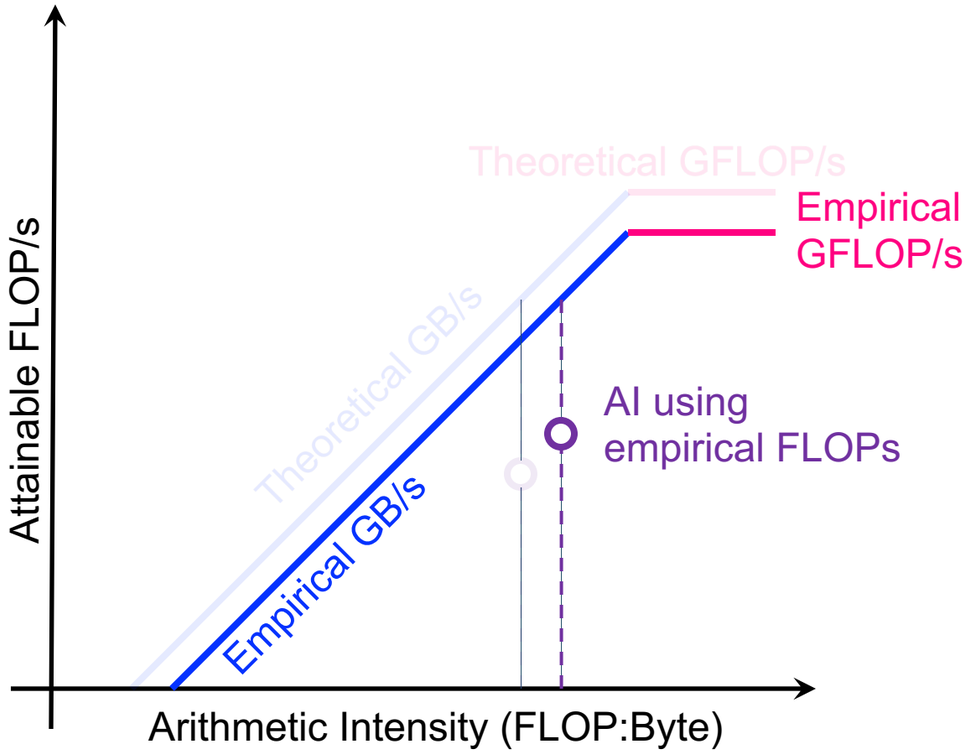
Theoretical vs. Empirical (Machine)

- Theoretical Roofline:
  - Pin bandwidth
  - FPUs * GHz
  - 1 C++ FLOP = 1 ISA FLOP
  - Data movement = Compulsory Misses
- Empirical Roofline:
  - Measured bandwidth
  - Measured Peak FLOP/s



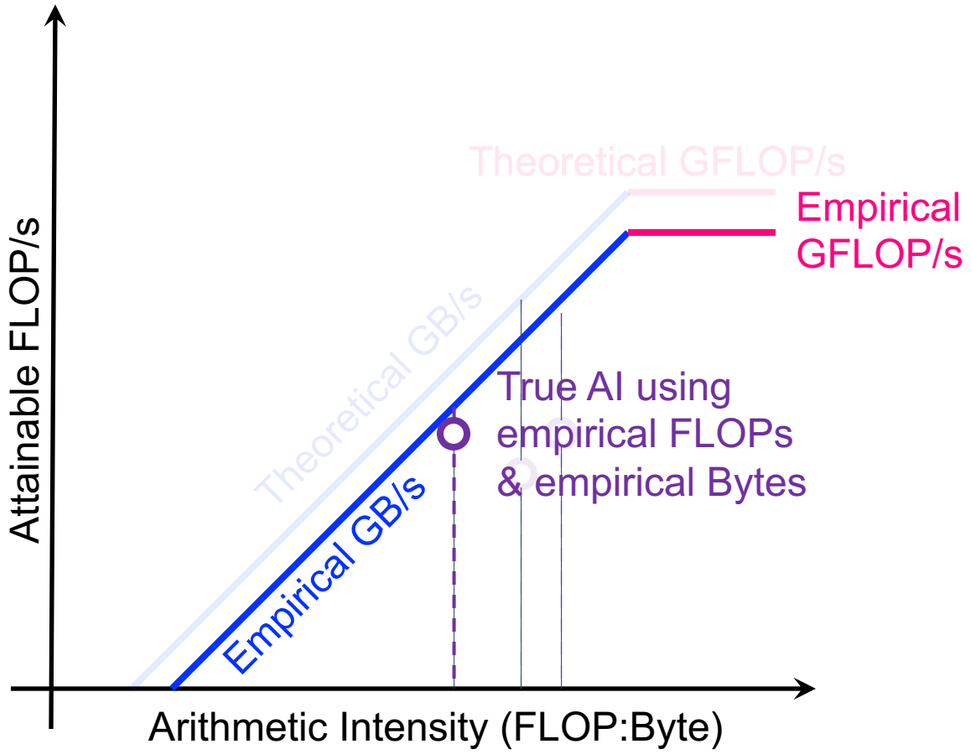
Theoretical vs. Empirical (Application FLOPs)

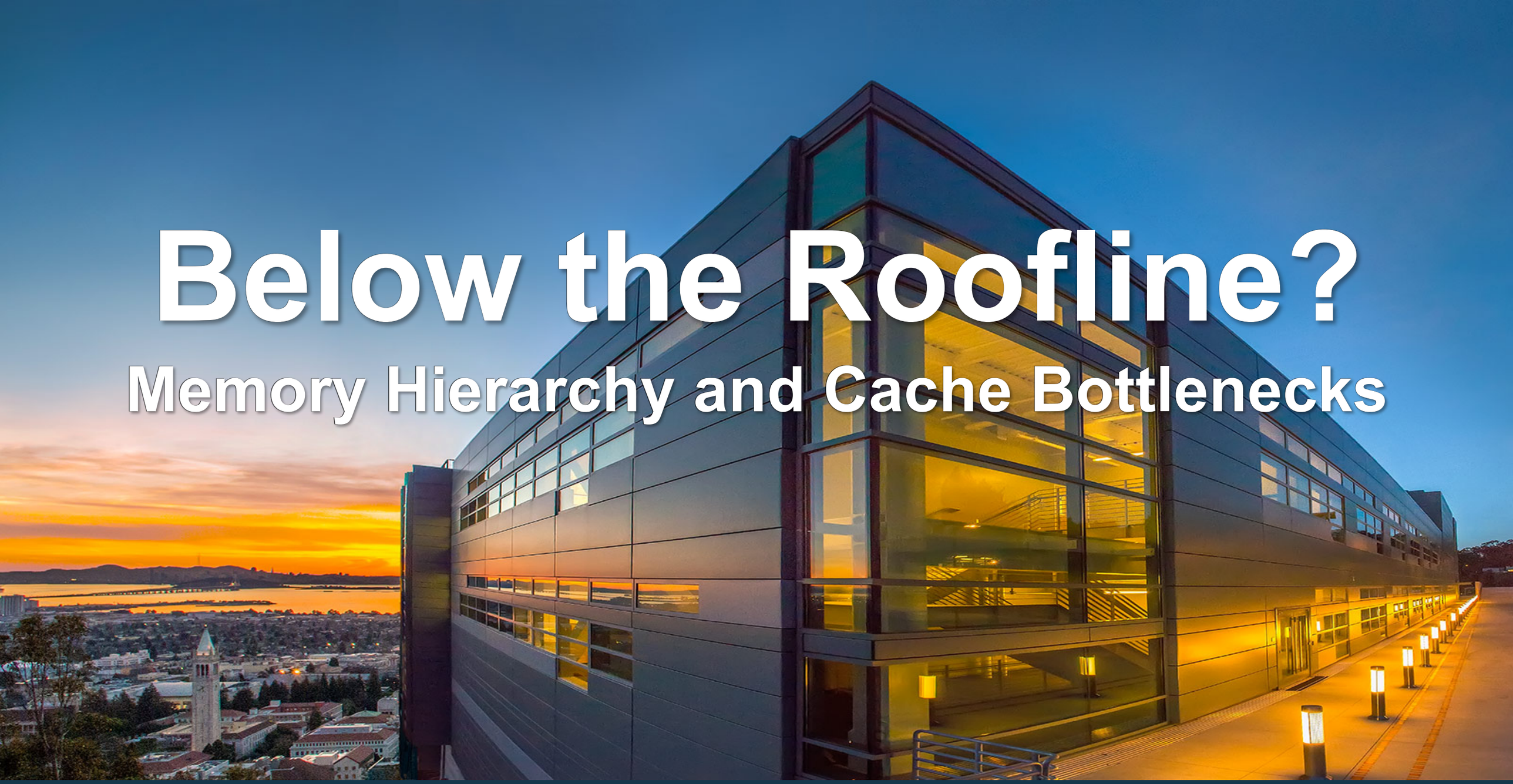
- Theoretical Roofline:
  - Pin bandwidth
  - FPUs * GHz
  - 1 C++ FLOP = 1 ISA FLOP
  - Data movement = Compulsory Misses
- Empirical Roofline:
  - Measured bandwidth
  - Measured Peak FLOP/s
  - 1 C++ FLOP >= 1 ISA FLOP (e.g. divide)



Theoretical vs. Empirical (Application Bytes)

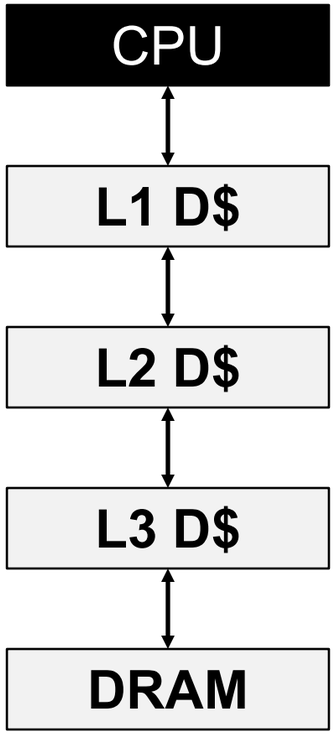
- Theoretical Roofline:
  - Pin bandwidth
  - FPUs * GHz
  - 1 C++ FLOP = 1 ISA FLOP
  - Data movement = Compulsory Misses
- Empirical Roofline:
  - Measured bandwidth
  - Measured Peak FLOP/s
  - 1 C++ FLOP >= 1 ISA FLOP (e.g. divide)
  - Measured data movement (cache effects)
  - ➢ **True Arithmetic Intensity can be higher or lower than expected**




Below the Roofline?
Memory Hierarchy and Cache Bottlenecks

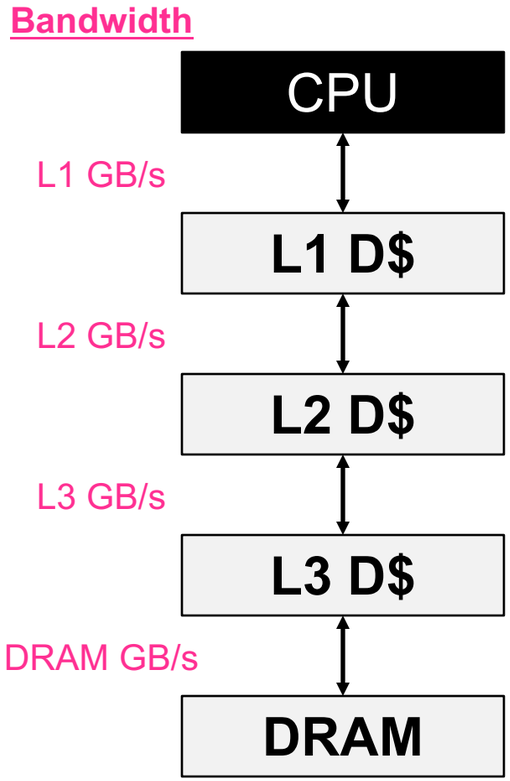
Memory Hierarchy

- Processors have multiple levels of memory/cache
  - Registers
  - L1, L2, L3 cache
  - HBM/HBM (KNL/GPU device memory)
  - DDR (main memory)
  - NVRAM (non-volatile memory)



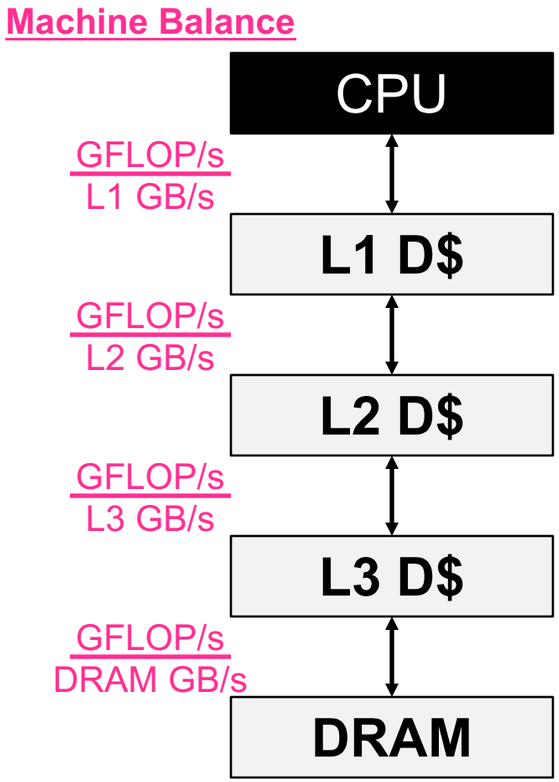
Memory Hierarchy

- Processors have different bandwidths for each level



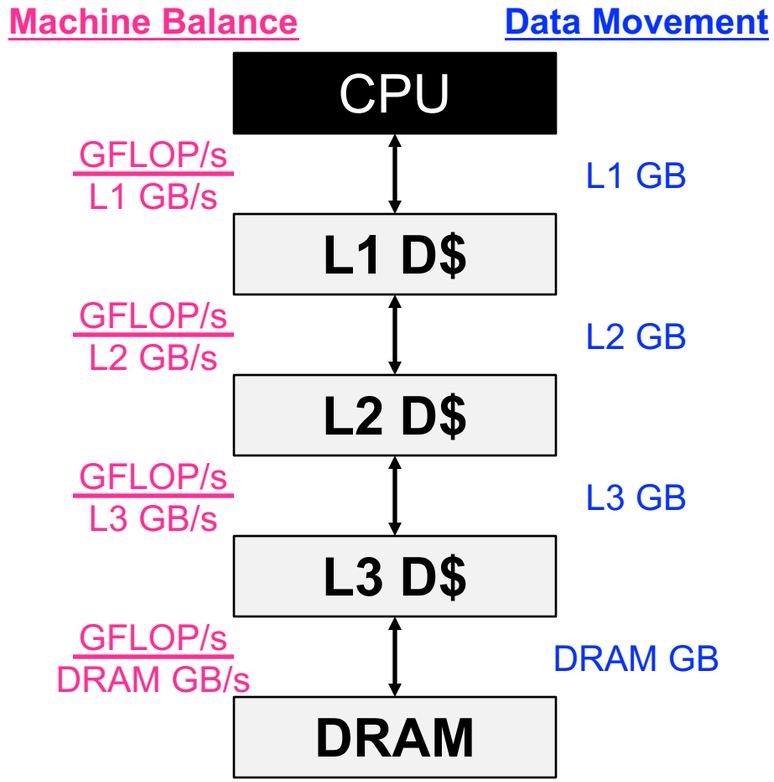
Memory Hierarchy

- Processors have different bandwidths for each level
  - different **machine balances** for each level



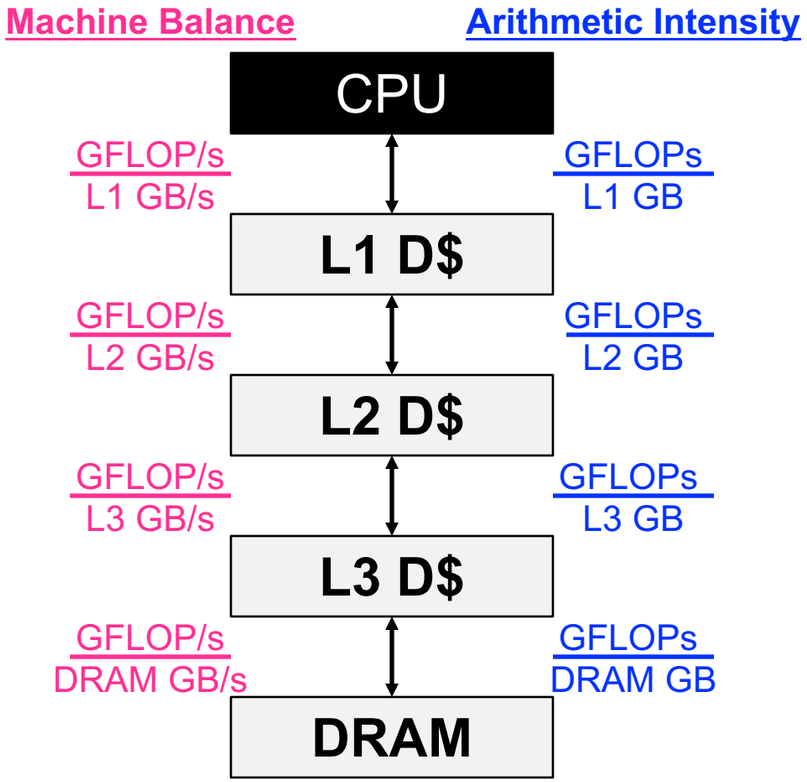
Memory Hierarchy

- Processors have different bandwidths for each level
  - different machine balances for each level
- Applications have locality in each level
  - different **data movements** for each level



Memory Hierarchy

- Processors have different bandwidths for each level
  - different machine balances for each level
- Applications have locality in each level
  - different data movements for each level
  - different **arithmetic intensity** for each level



Cache Bottlenecks

- For each additional level of the memory hierarchy, we can add another term to our model…

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI}_{\text{DRAM}} * \text{DRAM GB/s} \end{cases}$$

AI_x (Arithmetic Intensity at level “x”) = FLOPs / Bytes (moved to/from level “x”)

Cache Bottlenecks

- For each additional level of the memory hierarchy, we can add another term to our model…

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI}_{\text{DRAM}} * \text{DRAM GB/s} \\ \text{AI}_{\text{L2}} * \text{L2 GB/s} \end{cases}$$

AI_x (Arithmetic Intensity at level “x”) = FLOPs / Bytes (moved to/from level “x”)

Cache Bottlenecks

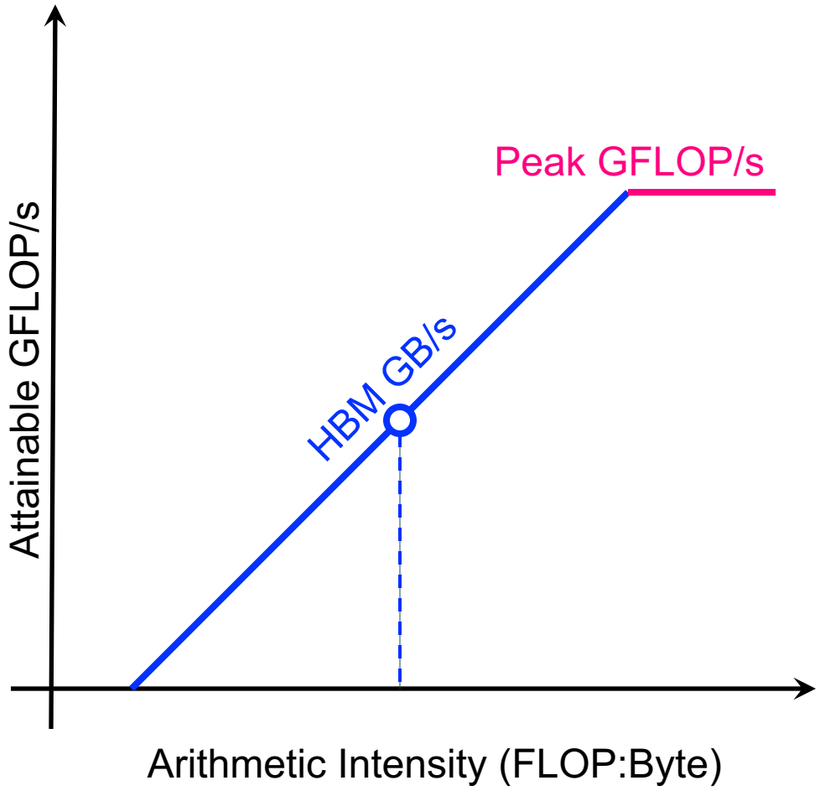
- For each additional level of the memory hierarchy, we can add another term to our model…

$$\text{GFLOP/s} = \min \begin{cases} \text{Peak GFLOP/s} \\ \text{AI}_{\text{DRAM}} * \text{DRAM GB/s} \\ \text{AI}_{\text{L2}} * \text{L2 GB/s} \\ \text{AI}_{\text{L1}} * \text{L1 GB/s} \end{cases}$$

AI_x (Arithmetic Intensity at level "x") = FLOPs / Bytes (moved to/from level "x")

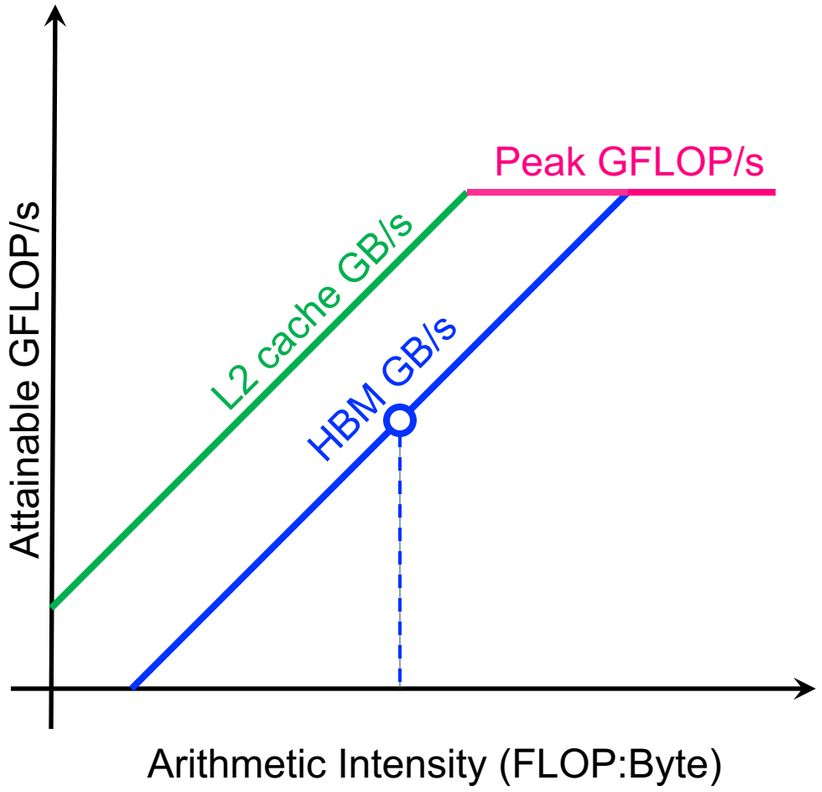
Cache Bottlenecks

- Plot equation in a single figure…
  - “**Hierarchical Roofline**” Model



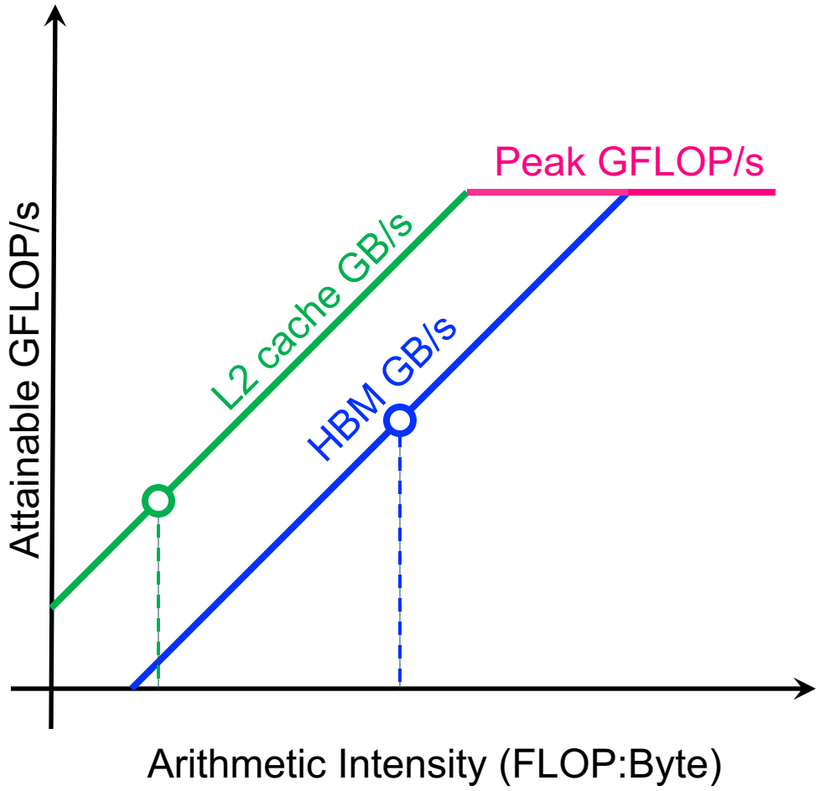
Cache Bottlenecks

- Plot equation in a single figure…
  - "**Hierarchical Roofline**" Model
  - Bandwidth ceiling (diagonal line) for each level of memory



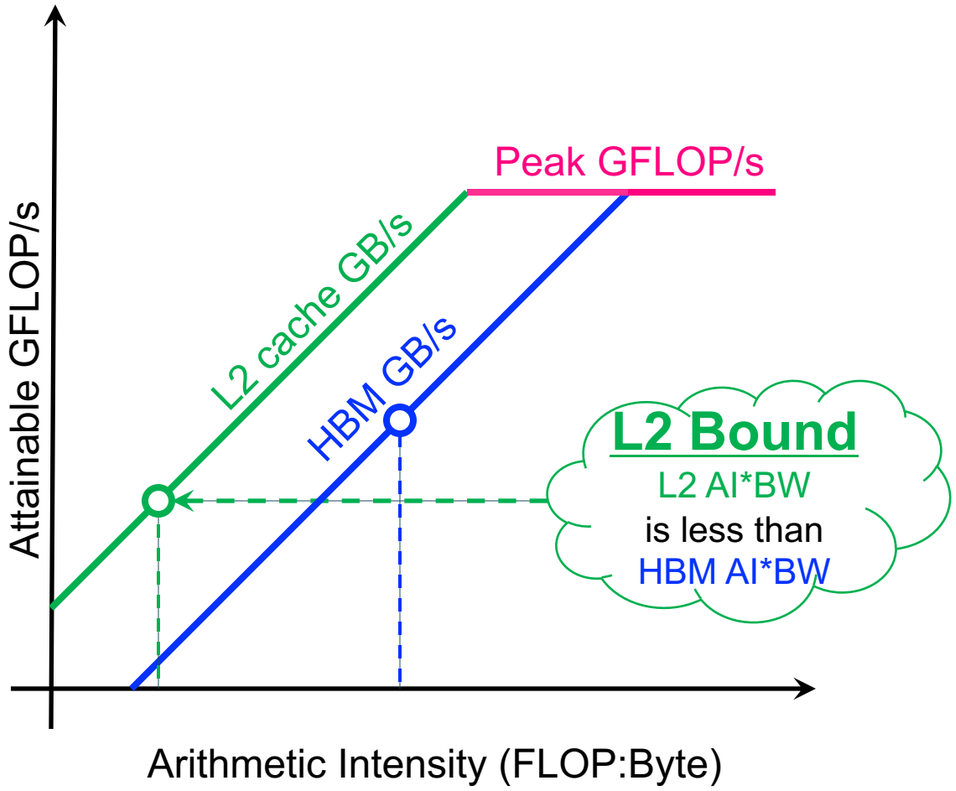
Cache Bottlenecks

- Plot equation in a single figure…
  - "**Hierarchical Roofline**" Model
  - Bandwidth ceiling (diagonal line) for each level of memory
  - Arithmetic Intensity (dot) for each level of memory



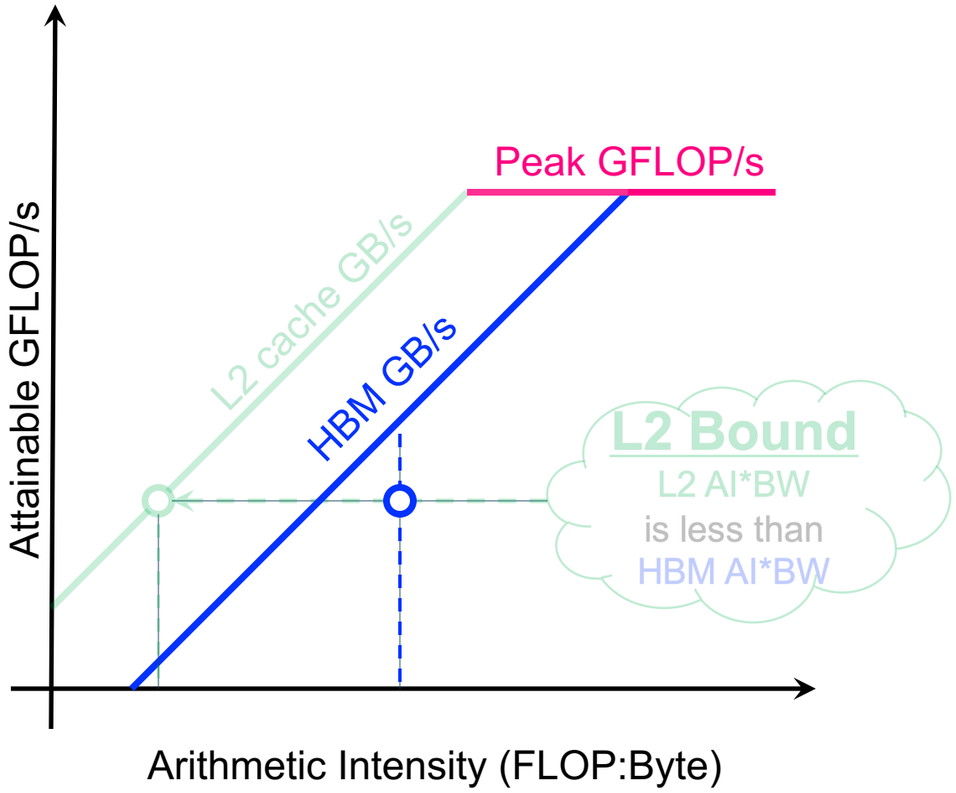
Cache Bottlenecks

- Plot equation in a single figure…
  - "**Hierarchical Roofline**" Model
  - Bandwidth ceiling (diagonal line) for each level of memory
  - Arithmetic Intensity (dot) for each level of memory
  - ➢ **performance is ultimately the minimum of these bounds**



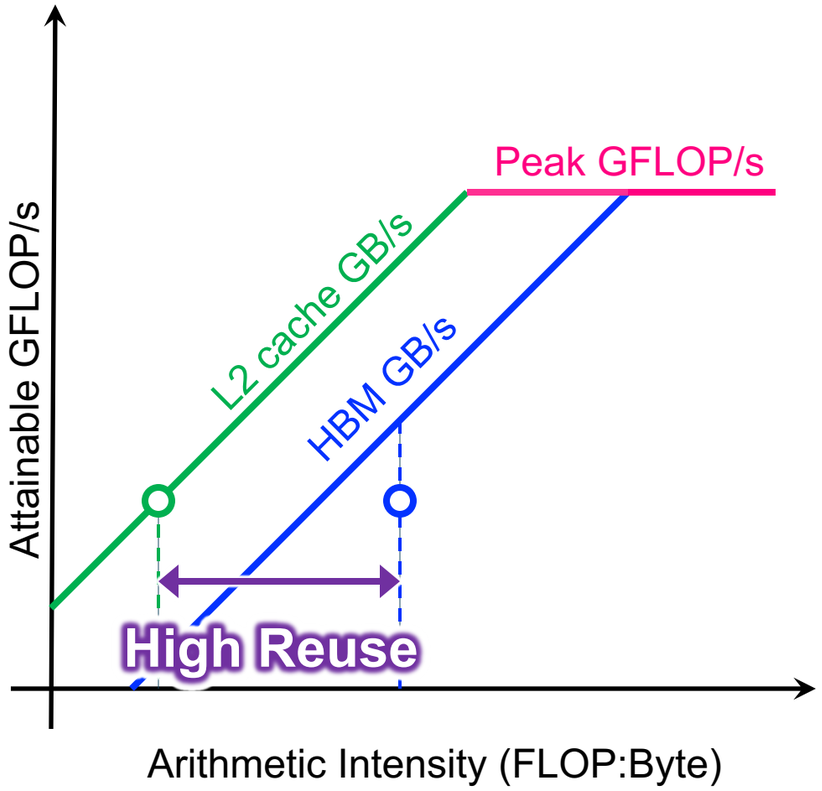
Cache Bottlenecks

- Plot equation in a single figure…
  - "**Hierarchical Roofline**" Model
  - Bandwidth ceiling (diagonal line) for each level of memory
  - Arithmetic Intensity (dot) for each level of memory
  - ➢ **performance is ultimately the minimum of these bounds**
- **If L2 bound, we see DRAM dot well below DRAM ceiling**



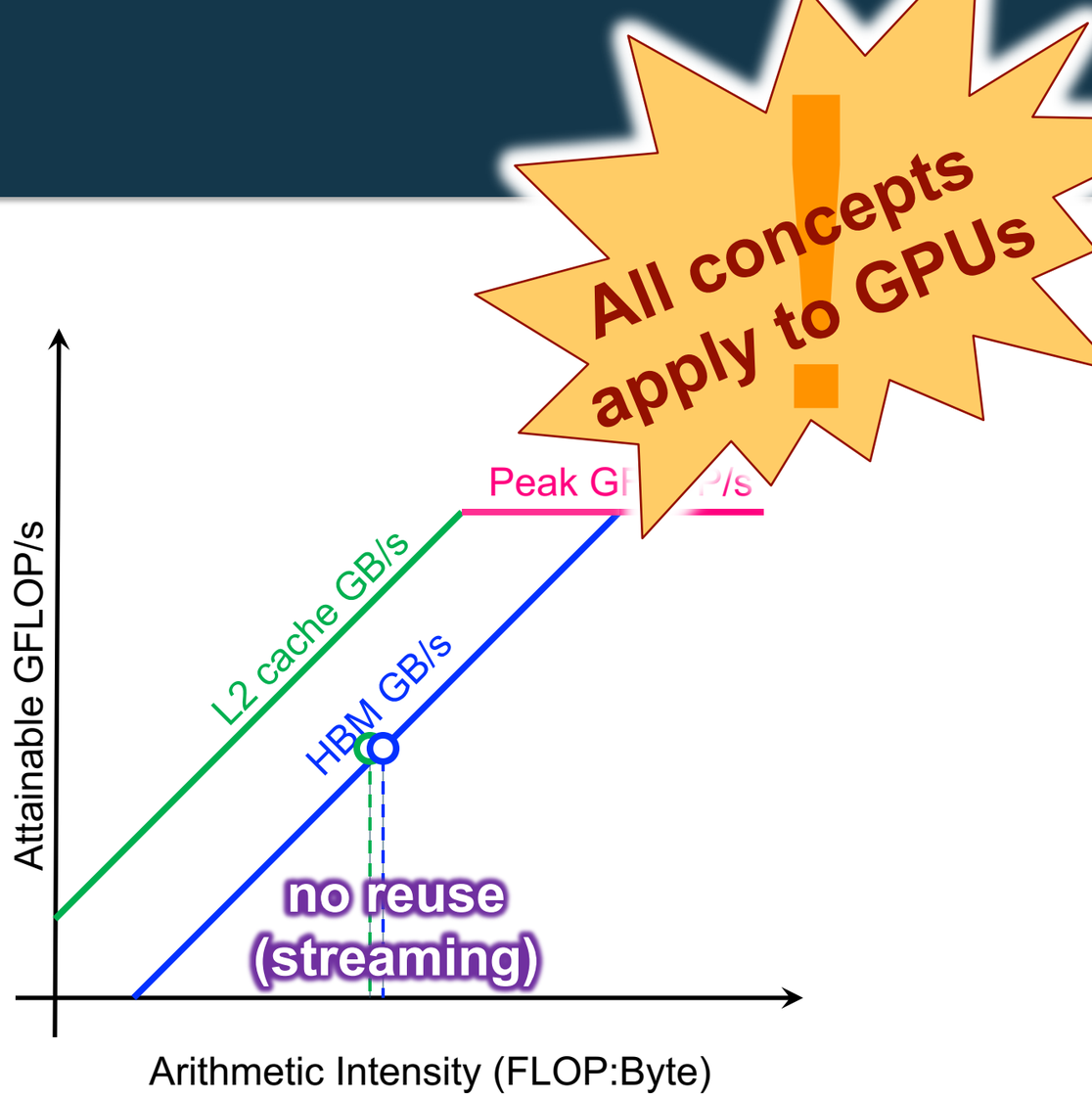
Cache Bottlenecks

- Widely separated Arithmetic Intensities indicate high reuse in the cache



Cache Bottlenecks

- Widely separated Arithmetic Intensities indicate high reuse in the cache
- Similar Arithmetic Intensities indicate effectively no cache reuse (**== streaming**)




Below the Roofline?
FMA, Vectorization, Tensor Cores
BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY
BERKELEY LAB
U.S. DEPARTMENT OF
ENERGY

Return of CISC

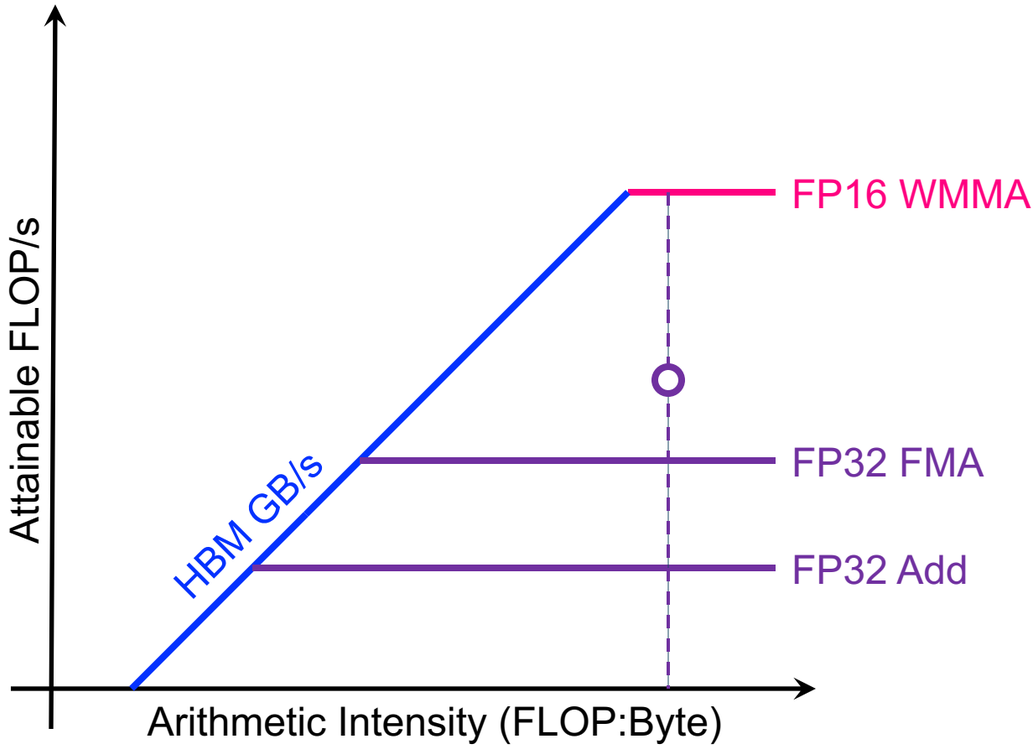
- Vectors have their limits (finite DLP, register file energy scales with VL, etc…)
- Death of Moore's Law is reinvigorating Complex Instruction Set Computing (CISC)

- Modern CPUs and GPUs are increasingly reliant on special (fused) instructions that perform multiple operations (fuse common instruction sequences)…
  - FMA (Fused Multiply Add): z=a*x+y …*z,x,y are vectors or scalars*
  - 4FMA (Quad FMA): z=A*x+z …*A is a FP32 matrix; x,z are vectors*
  - WMMA (Tensor Core): Z=AB+C …*A,B are FP16 matrices; Z,C are FP32*

➢ **If instructions are a mix or scalar (predicated), vector, and matrix operations, performance is now a weighted average of them.**

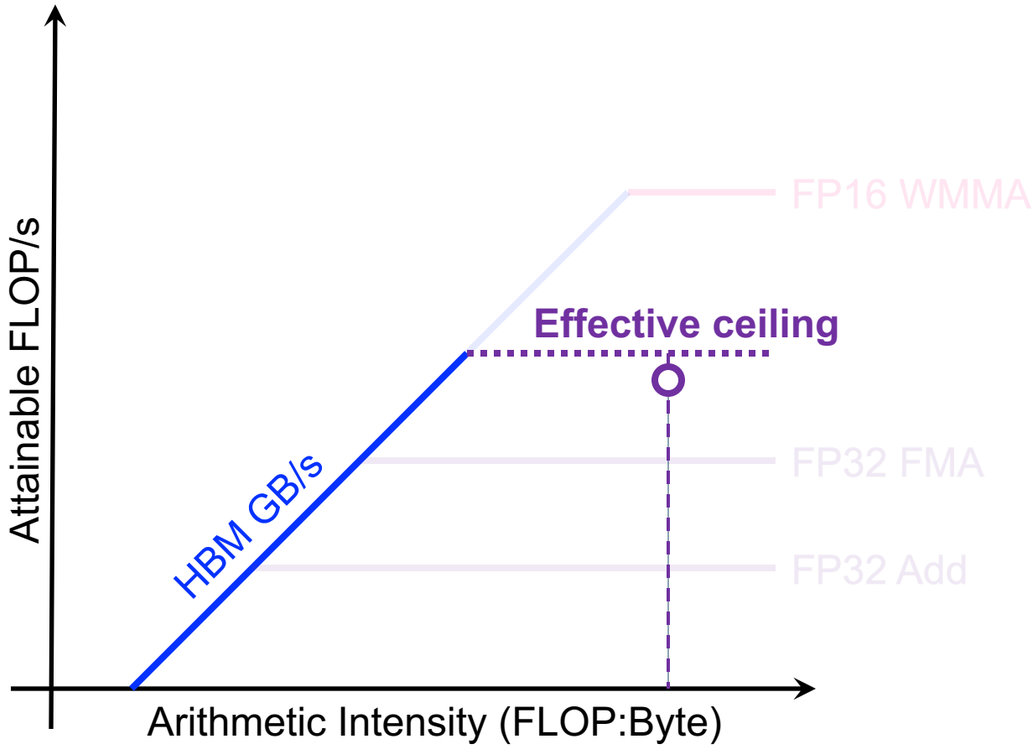
Return of CISC

- Consider NVIDIA Volta GPU…
  - ~100 TFLOPs for FP16 Tensor
  - 15 TFLOPS for FP32 FMA
  - 7.5 TFLOPs for FP32 Add
- DL applications mix Tensor, FP16, and FP32
- DL performance may be well below nominal Tensor Core peak



Return of CISC

- Consider NVIDIA Volta GPU…
  - ~100 TFLOPs for FP16 Tensor
  - 15 TFLOPS for FP32 FMA
  - 7.5 TFLOPs for FP32 Add
- DL applications mix Tensor, FP16, and FP32
- DL performance may be well below nominal Tensor Core peak
- The actual mix of instructions introduces an **effective ceiling** on performance…




Below the Roofline?
FPU Starvation

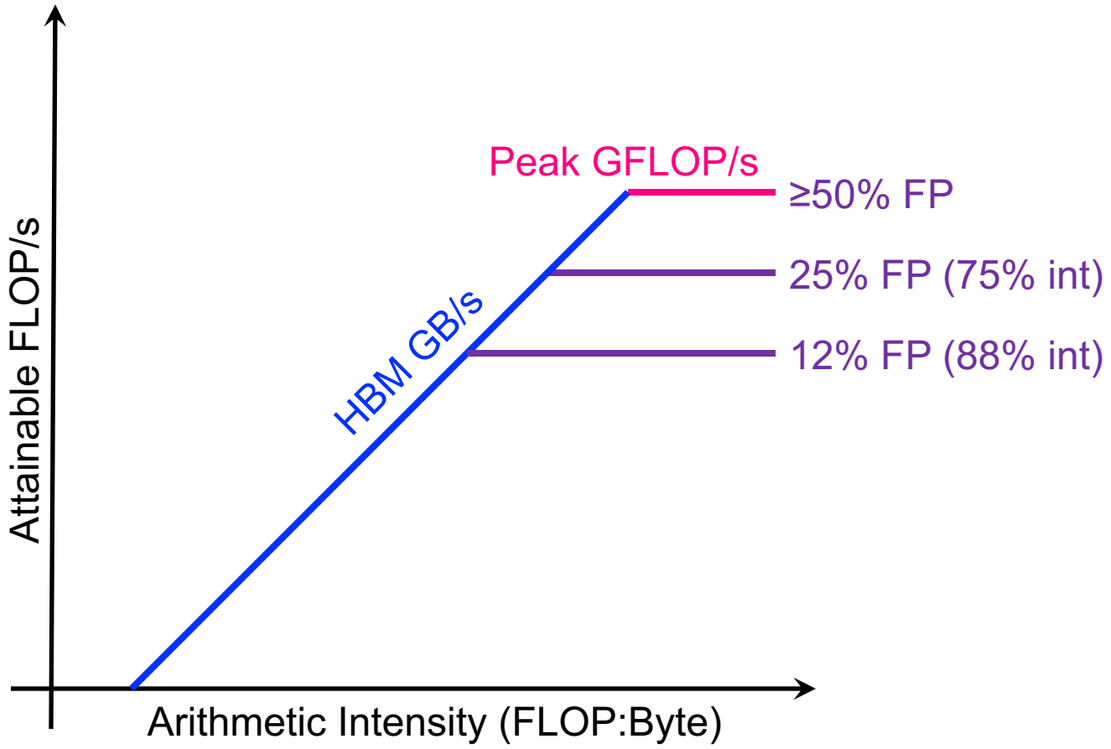
FPU Starvation

- Processors have finite instruction fetch/decode/issue bandwidth
- Moreover, the number of FP units dictates the FP issue rate required to hit peak

➢ **Ratio of these two rates is the minimum FP instruction fraction required to hit peak**

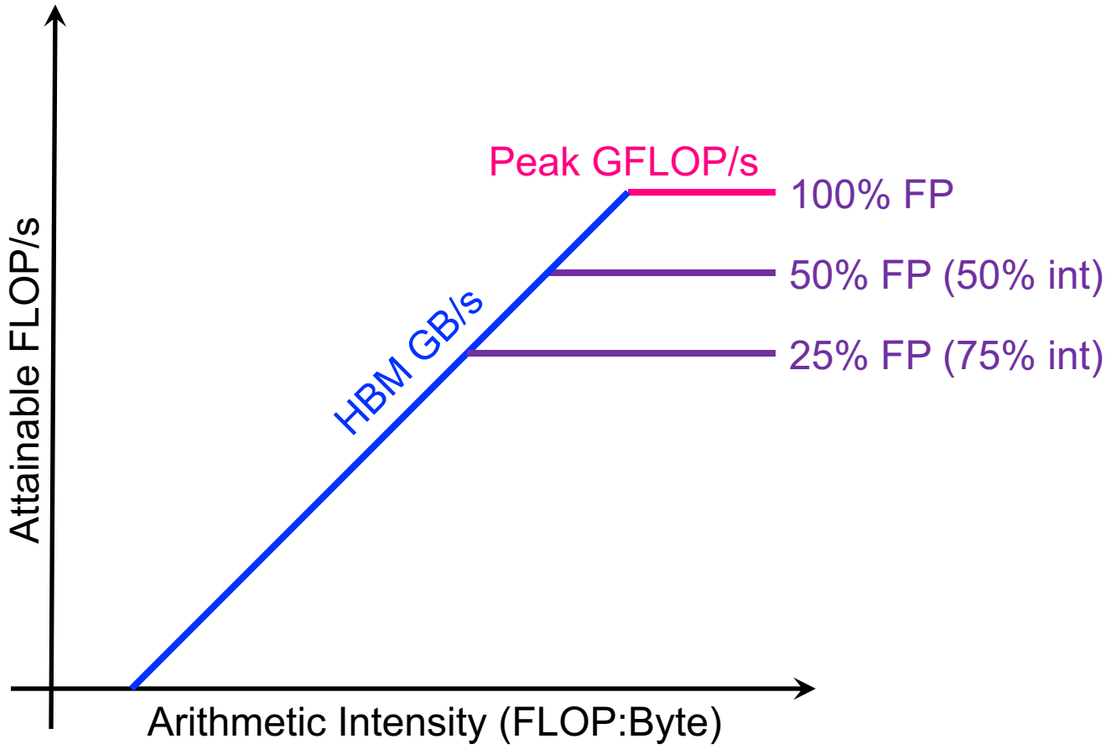
FPU Starvation

- Consider…
  - 4-issue superscalar
  - 2 FP data paths
  - ➢ **>50% of the instructions** must be FP to have any chance at peak performance



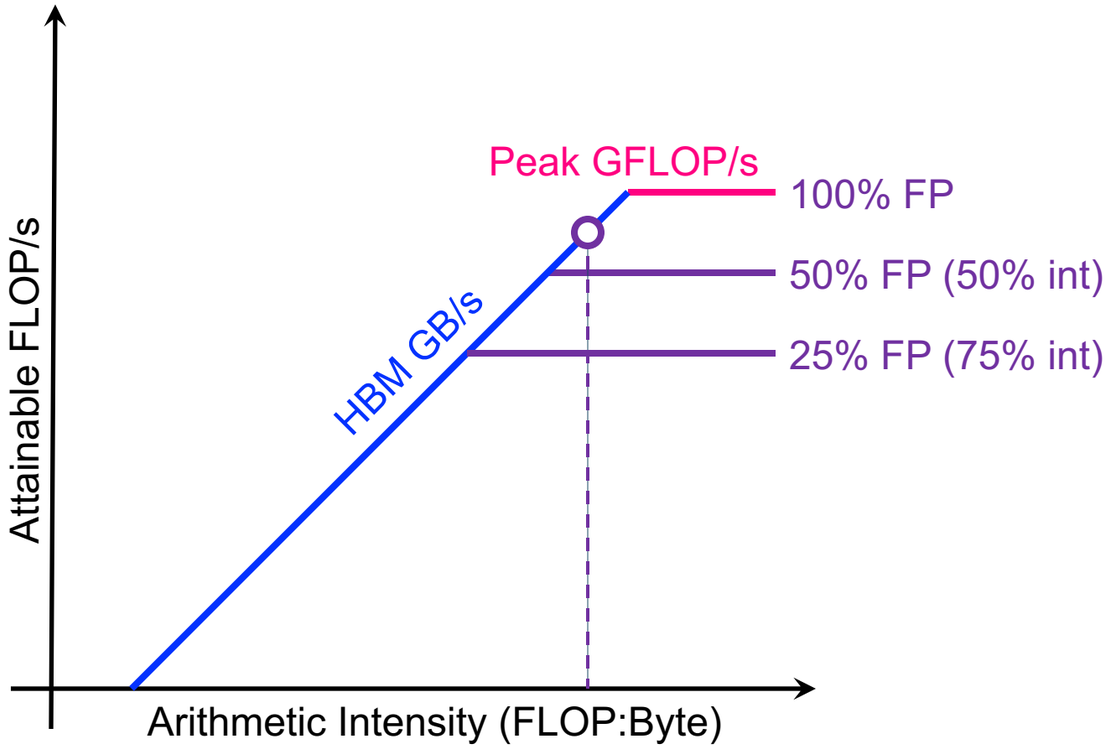
FPU Starvation

- Conversely,
  - Keeping 2 FP data paths,
  - but downscaling to 2-issue superscalar
  - ➢ **100% of the instructions must be FP to get peak performance**



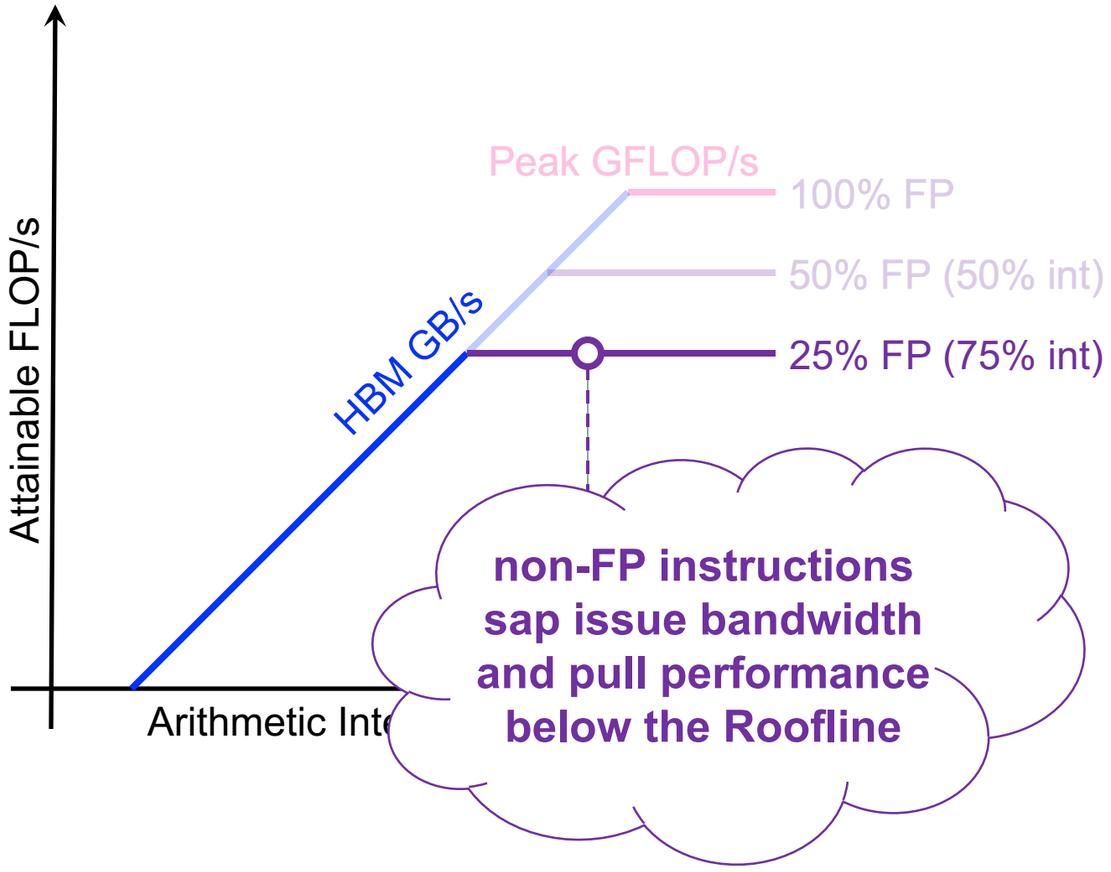
FPU Starvation

- Conversely,
  - Keeping 2 FP data paths,
  - but downscaling to 2-issue superscalar
  - ➢ **100% of the instructions must be FP to get peak performance**



FPU Starvation

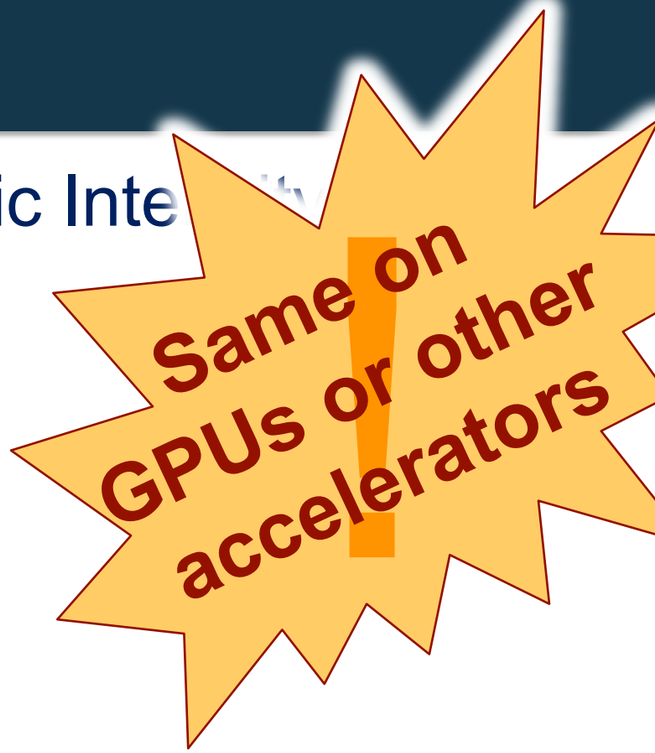
- Conversely,
  - Keeping 2 FP data paths,
  - but downscaling to 2-issue superscalar
  - 100% of the instructions must be FP to get peak performance
  - ➢ **Codes that would have been memory-bound are now decode/issue-bound.**




Recap

Recap

- Roofline bounds performance as a function of Arithmetic Inte
  - Horizontal Lines = Compute Ceilings
  - Diagonal Lines = Bandwidth Ceilings
  - Bandwidth ceilings are parallel on log-log scale
  - ➢ **Collectively, ceilings define an upper limit on performance**
- Loop Arithmetic Intensity (for each level of memory)
  - Total FLOPs / Total Data Movement
  - Includes **all** cache effects
  - ➢ **Measure of a loop's temporal locality**
- Plotting loops on the Roofline
  - Each loop has one dot per level of memory
  - x-coordinate = arithmetic intensity at that level
  - y-coordinate = performance (e.g. GFLOP/s)
  - ➢ **Proximity to associated ceiling is indicative of a performance bound**
  - ➢ **Position of dots relative to each other is indicative of cache locality**

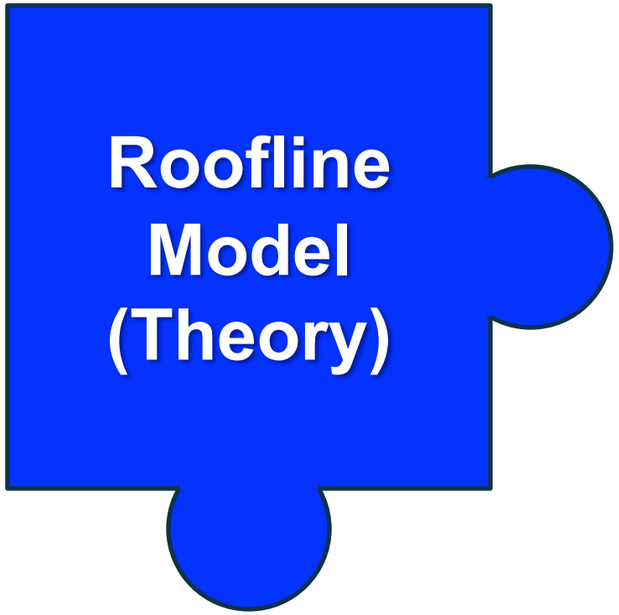


What is Roofline used for?

- Understand performance differences between Architectures, Programming Models, implementations, etc…
  - Why do some Architectures/Implementations move more data than others?
  - Why do some compilers outperform others?
- Predict performance on future machines / architectures
  - Set realistic performance expectations
  - Drive for HW/SW Co-Design
- Identify performance bottlenecks & motivate software optimizations
- Determine when we're done optimizing code
  - Assess performance relative to machine capabilities
  - Track progress towards optimality
  - Motivate need for algorithmic changes

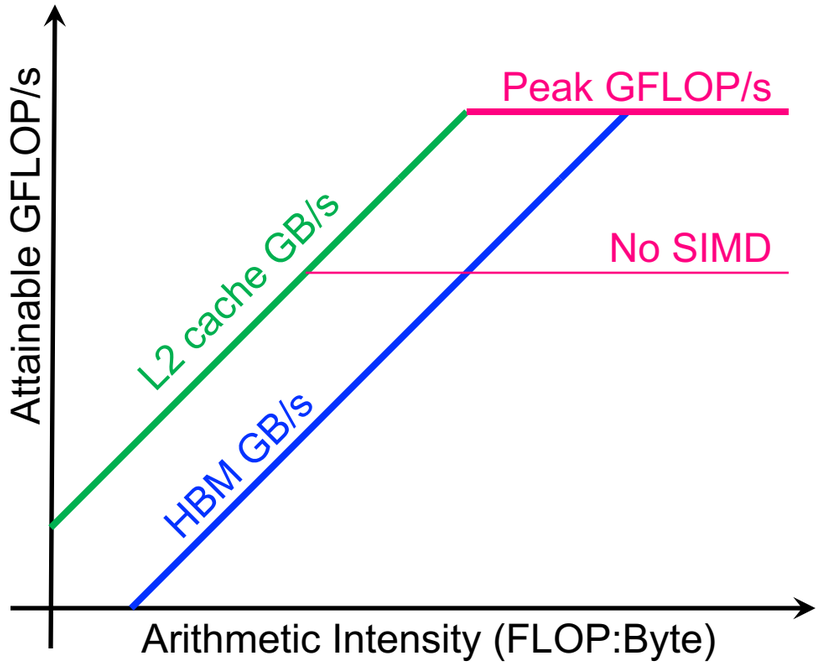
Model is just one piece of the puzzle…

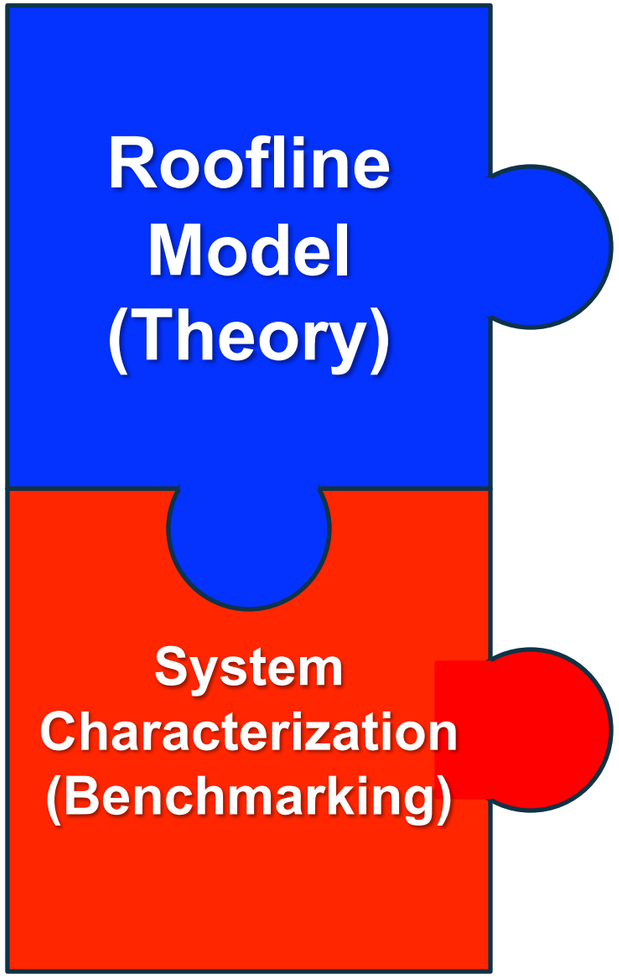
- Roofline Model defines the basic concepts and equations.



Model is just one piece of the puzzle…

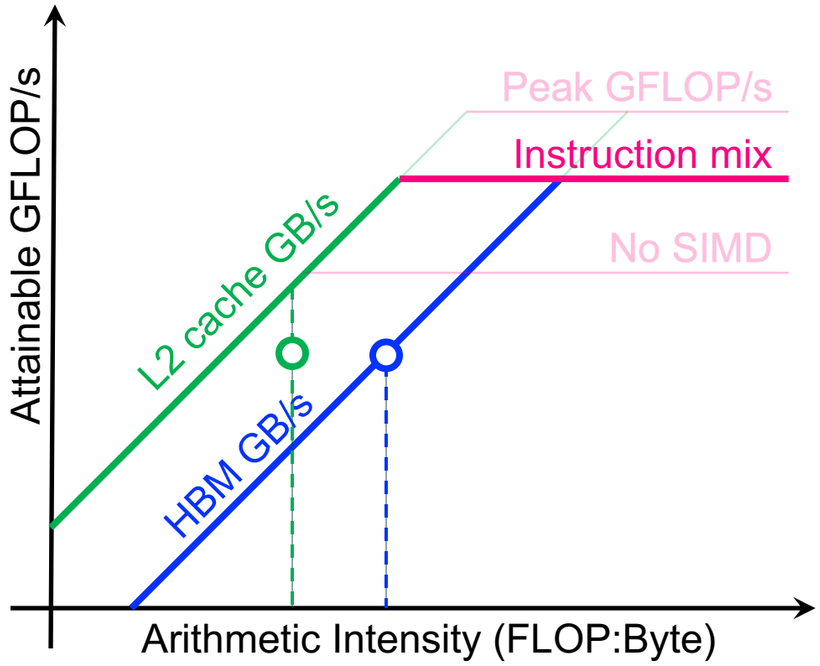
- System Characterization defines the shape of the Roofline (peak bandwidths and FLOP/s)

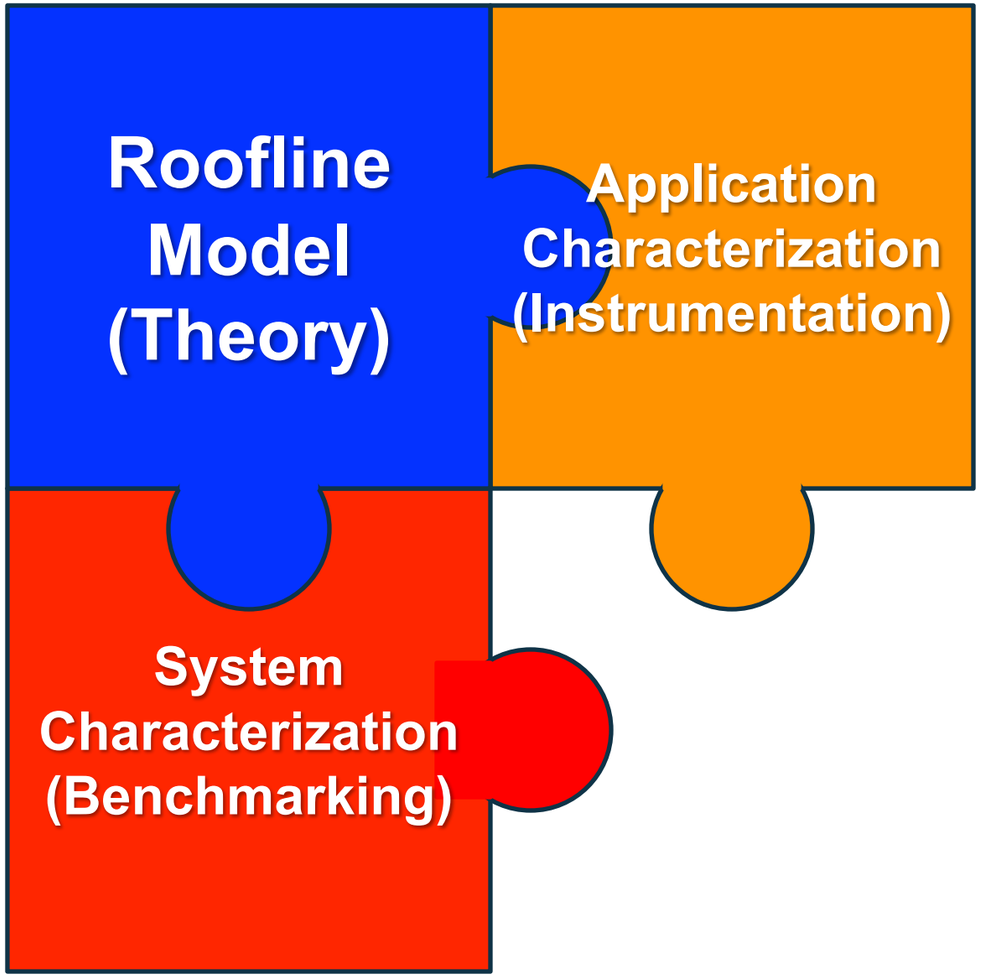




Model is just one piece of the puzzle…

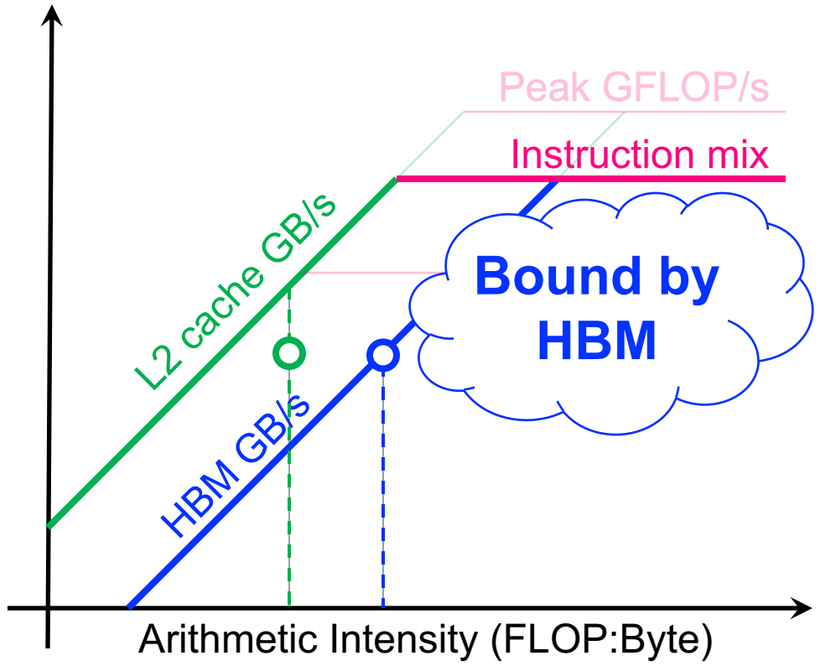
- Application Characterization determines…
  - Intensity and Performance of each loop
  - Position of any implicit ceilings

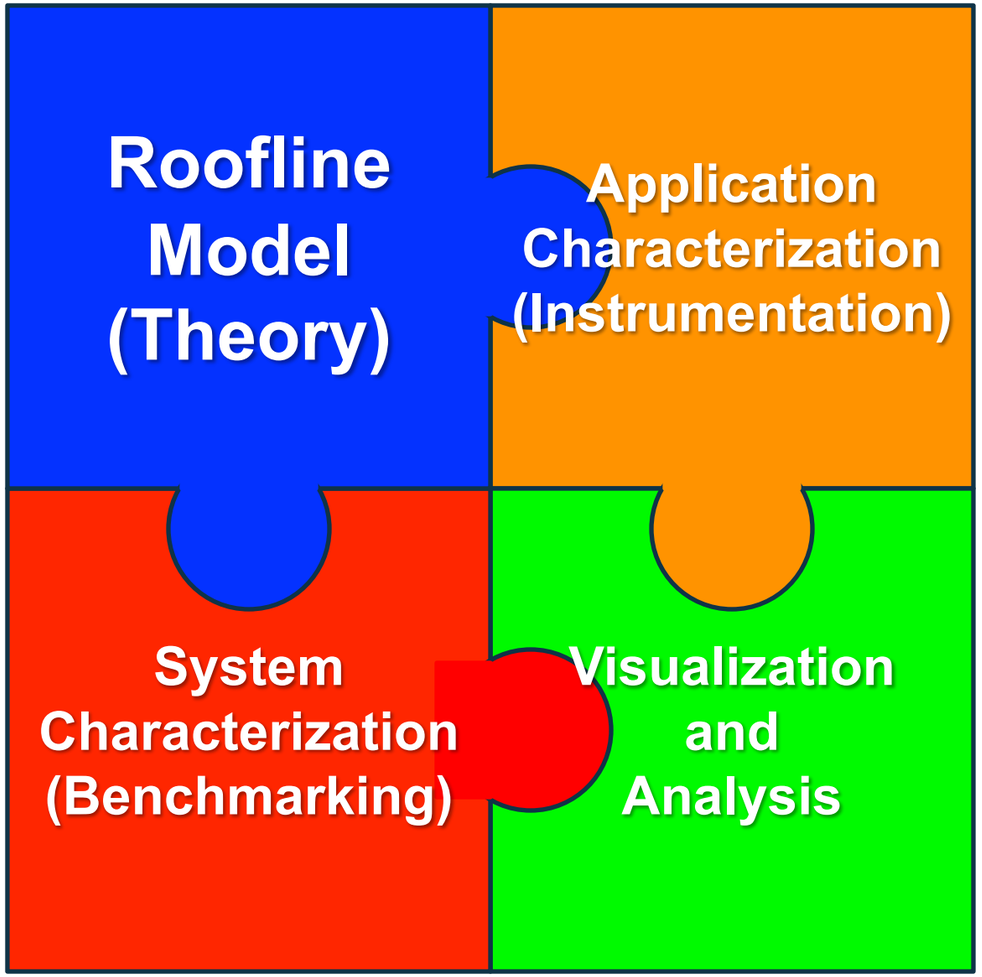




Model is just one piece of the puzzle…

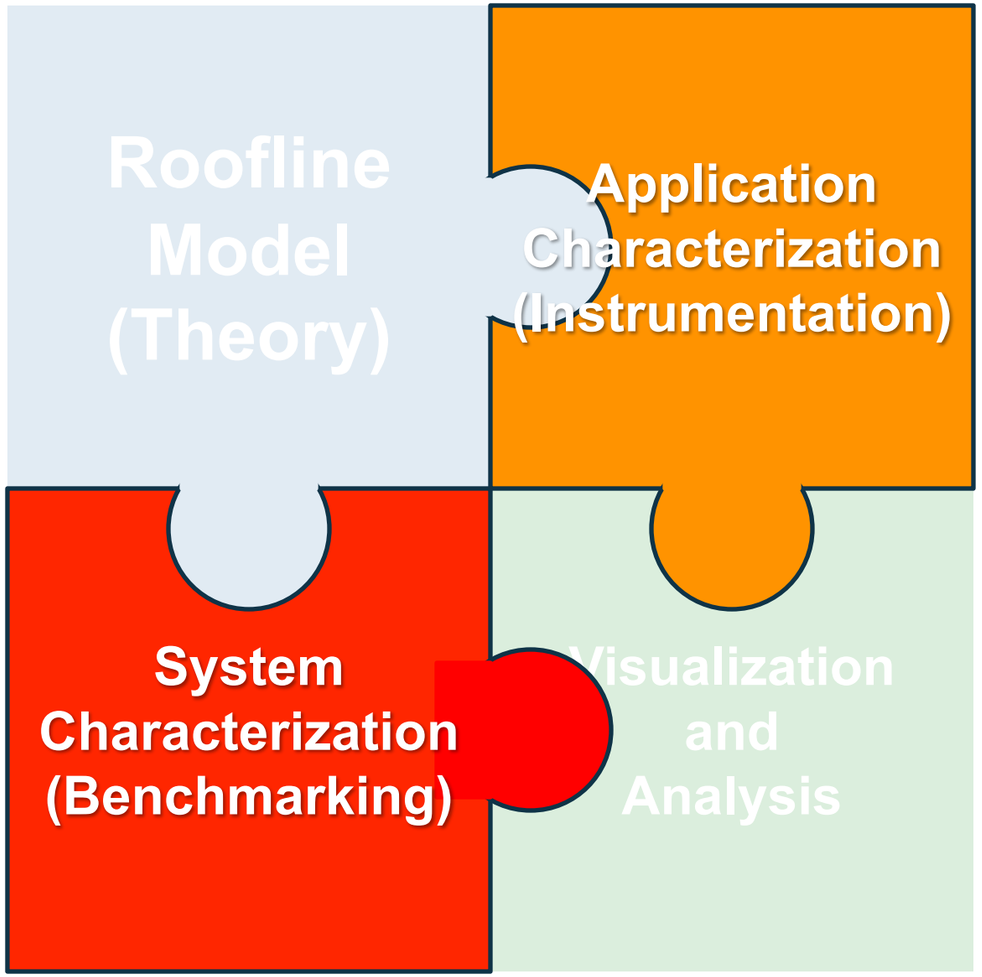
- Visualization tools combine all data together and provide analytical capability





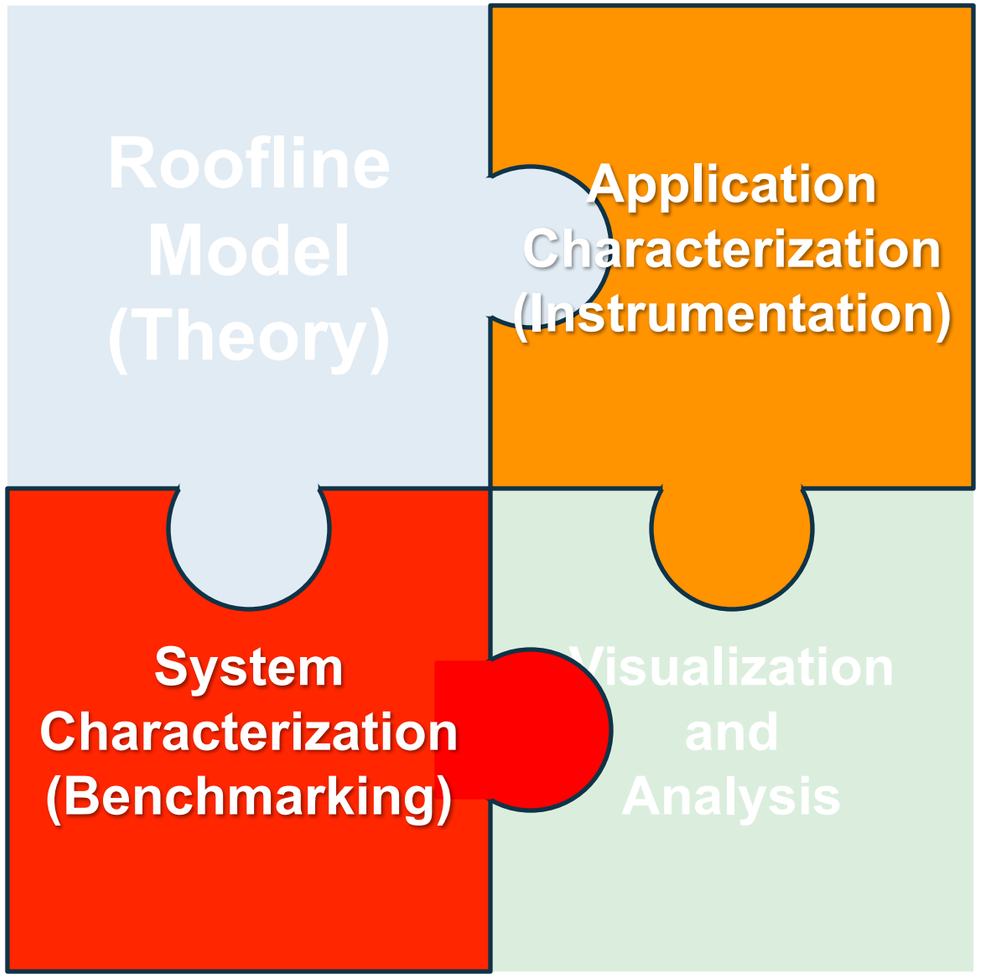
Rest of Tutorial…

- Charlene will demonstrate how to apply the Roofline model to NVIDIA GPUs
  - GPU benchmarking
  - application characterization



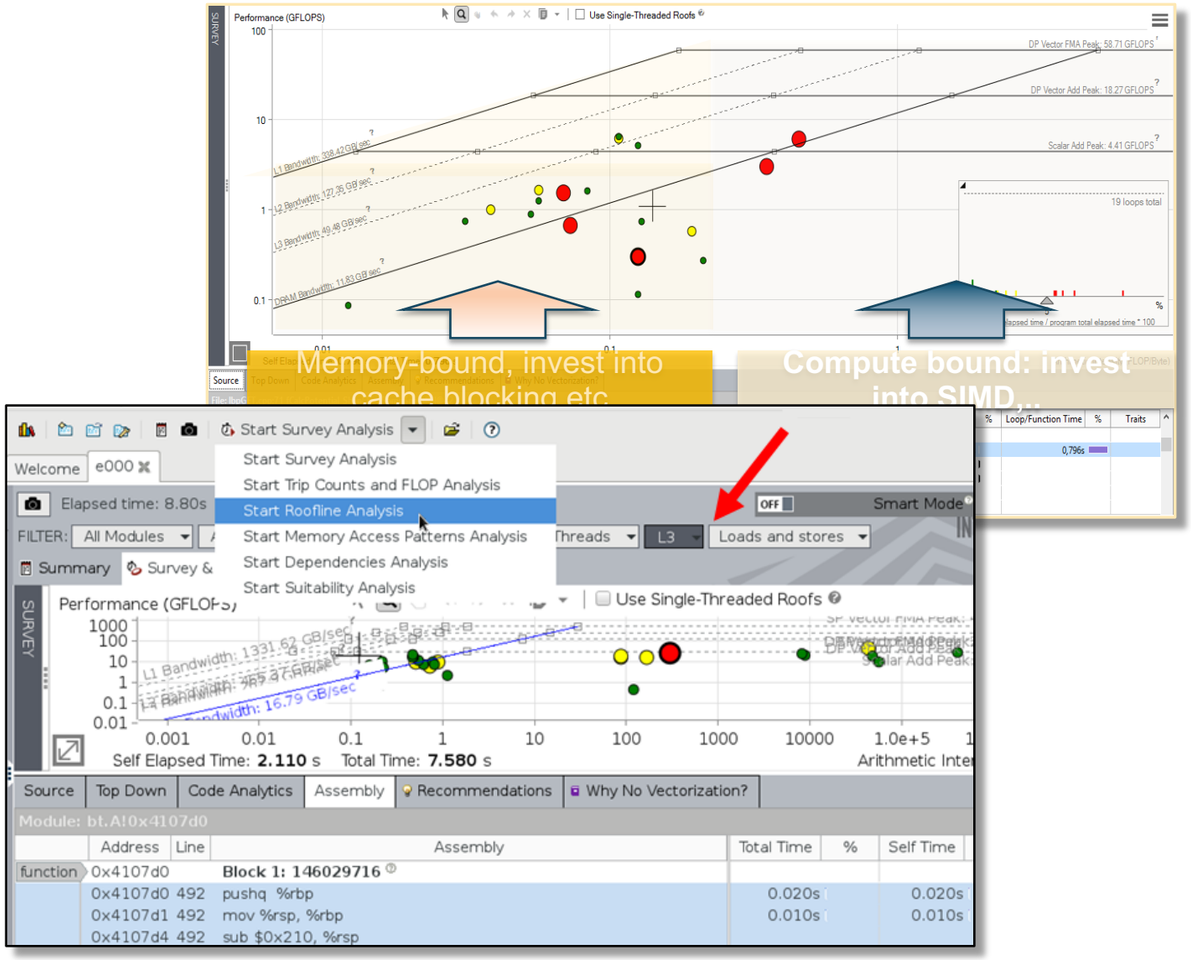
Rest of Tutorial…

- Charlene will demonstrate how to apply the Roofline model to NVIDIA GPUs
  - GPU benchmarking
  - application characterization
- Sam will extend this by examining advanced GPU topics…
  - Instruction Roofline Model
  - Using Roofline to analyze DL codes
  - Scaling Trajectories



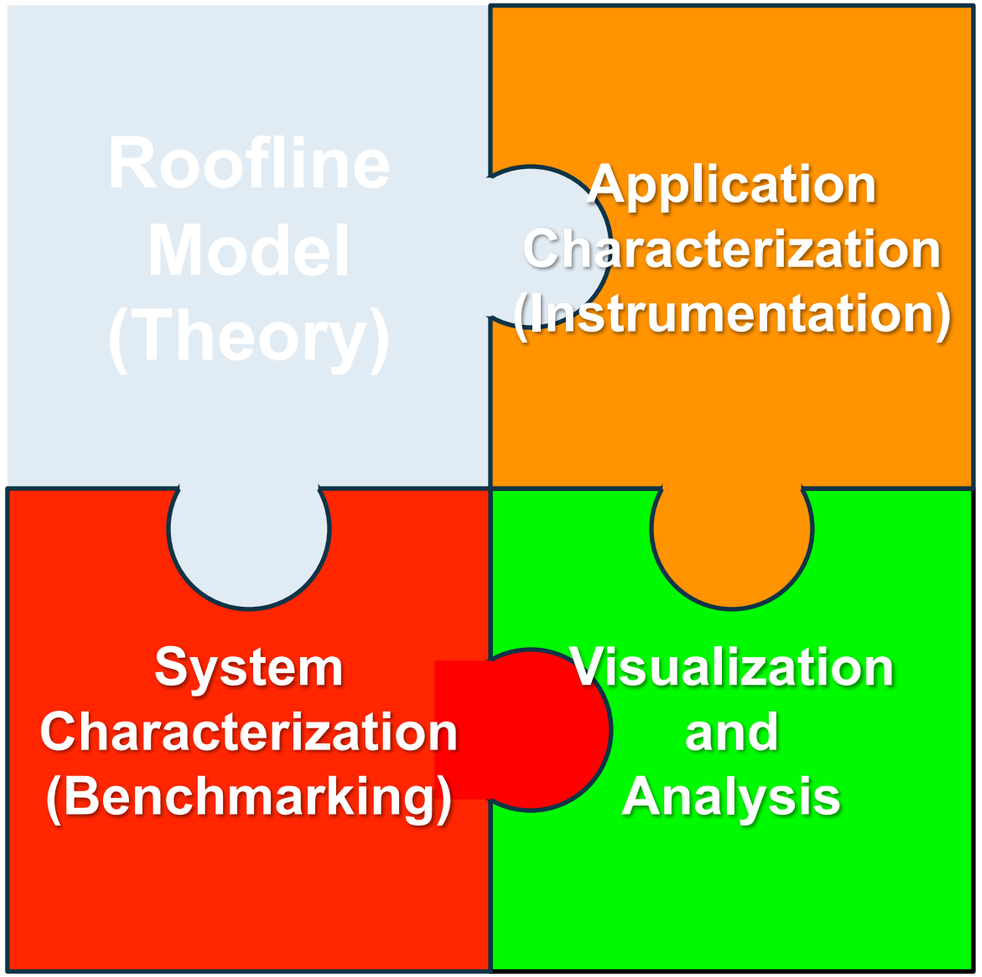
Rest of Tutorial…

- Charlene will introduce and demo **Intel® Advisor**
  - ✓ Automatically instruments applications (one dot per loop nest/function)
  - ✓ Computes FLOPS and AI for each function (**CARM**)
  - ✓ **Integrated Cache Simulator (hierarchical roofline / multiple AI's)**
  - ✓ AVX-512 support that incorporates masks
  - ✓ Automatically benchmarks target system (calculates ceilings)
  - ✓ Full integration with existing Advisor capabilities



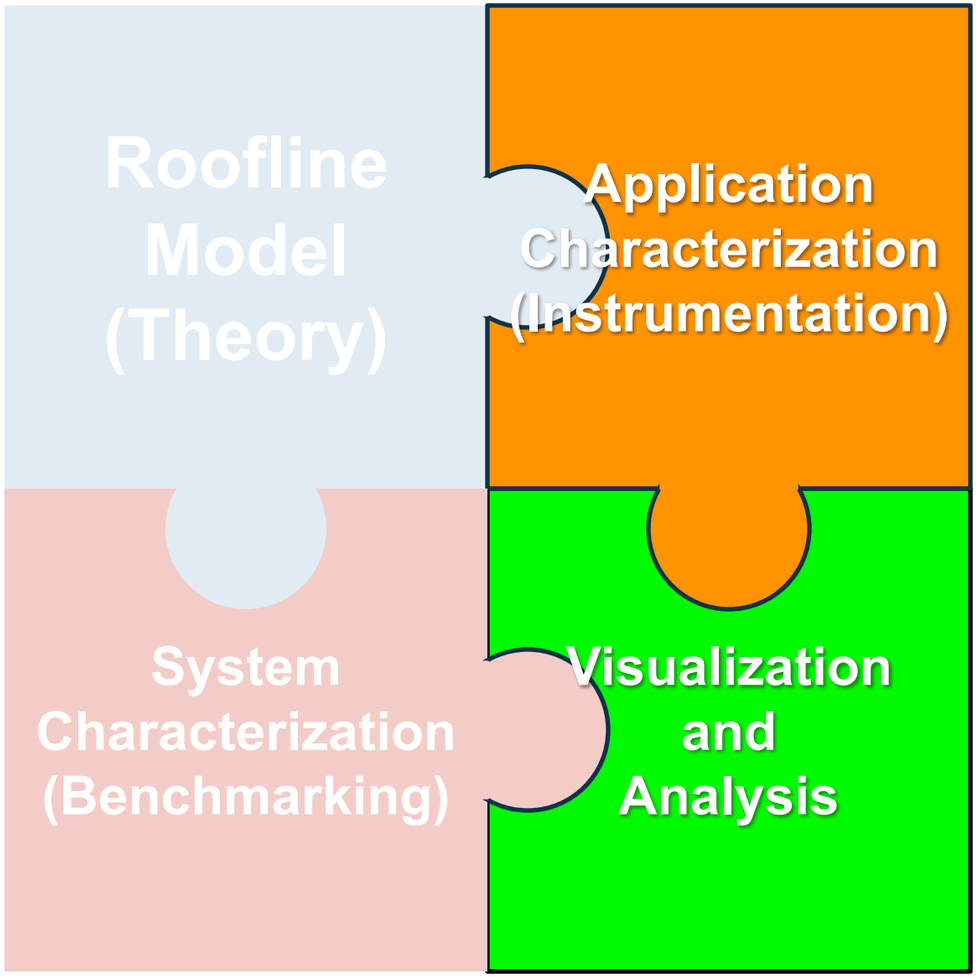
Rest of Tutorial…

- Charlene will introduce and demo **Intel® Advisor**
  - ✓ Automatically instruments applications (one dot per loop nest/function)
  - ✓ Computes FLOPS and AI for each function (**CARM**)
  - ✓ **Integrated Cache Simulator (hierarchical roofline / multiple AI's)**
  - ✓ AVX-512 support that incorporates masks
  - ✓ Automatically benchmarks target system (calculates ceilings)
  - ✓ Full integration with existing Advisor capabilities



Rest of Tutorial…

- Jack will discuss how Roofline is used by the NERSC NESAP teams



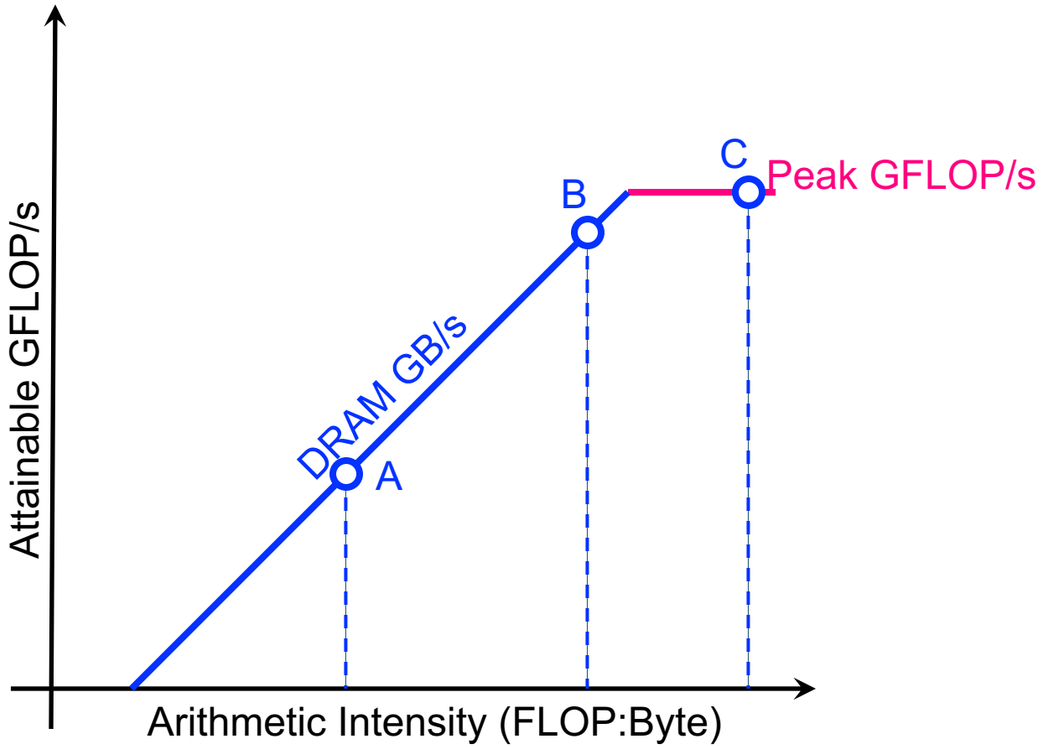

Questions?


BACKUP


Performance
Extrapolations

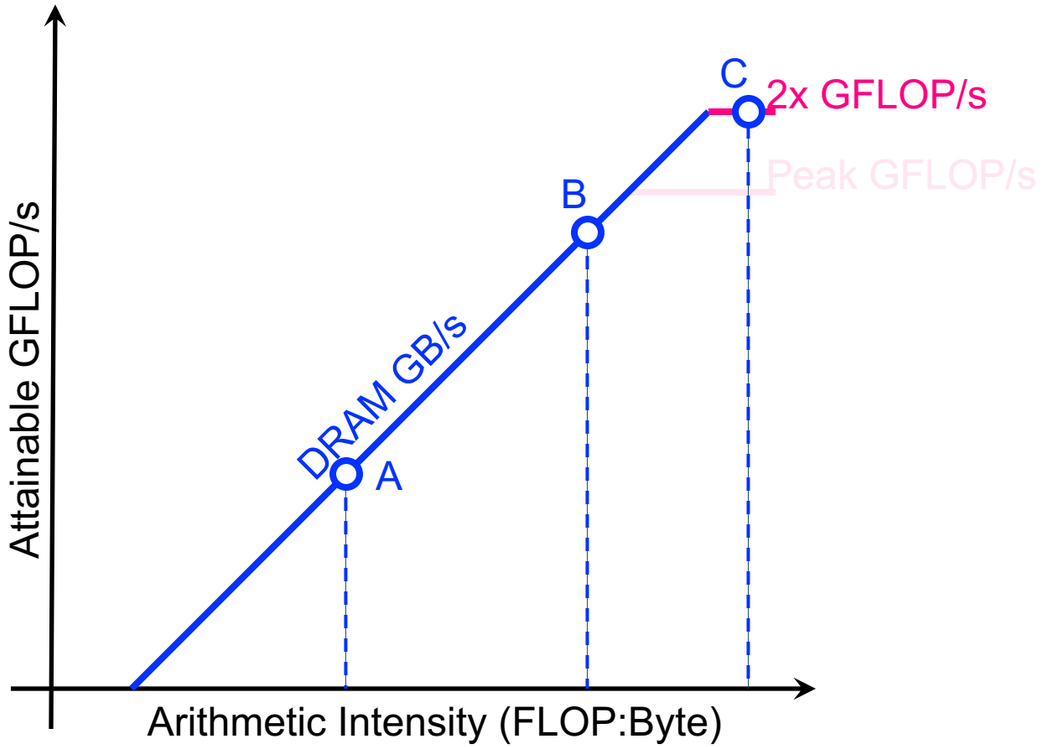
Setting Realistic Expectations…

- Consider 3 kernels (A,B,C)
  - **kernels A and B are bound by memory bandwidth**
  - **kernel C is bound by peak FLOP/s**



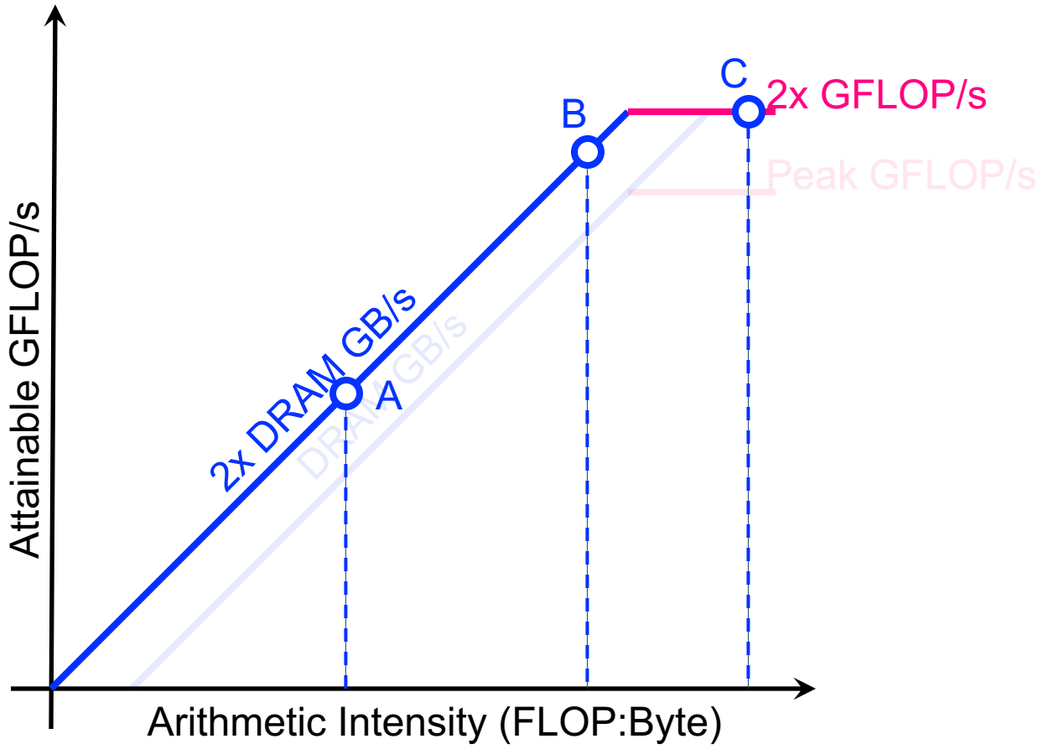
Setting Realistic Expectations…

- Imagine you want to run on a machine with twice the peak FLOPs…
  - ➢ **kernel C's performance could double**
  - ✗ **kernels A and B will be no faster**



Setting Realistic Expectations…

- What if that machine also doubled memory bandwidth…
  - **kernel A and B's performance could also double**




Retrospective

Performance Model Retrospective

- Too many components contribute to app/loop run time…
  - some are characteristics of the application
  - some are characteristics of the machine
  - some are both (memory access pattern + caches)

#FP operations	FLOP/s
Cache data movement	Cache GB/s
DRAM data movement	DRAM GB/s
PCIe data movement	PCIe bandwidth
Depth	OMP Overhead
MPI Message Size	Network Bandwidth
MPI Send:Wait ratio	Network Gap
#MPI Wait's	Network Latency

Performance Model Retrospective

- Performance models often conceptualize the system as being dominated by one or more aspects of machine and application…

#FP operations	FLOP/s
Cache data movement	Cache GB/s
DRAM data movement	DRAM GB/s
PCIe data movement	PCIe bandwidth
Computational Depth	OMP Overhead
MPI Message Size	Network Bandwidth
MPI Send:Wait ratio	Network Gap
#MPI Wait's	Network Latency

Performance Model Retrospective

- Performance models often conceptualize the system as being dominated by one or more aspects of machine and application…

#FP operations	FLOP/s
Cache data movement	Cache GB/s
DRAM data movement	DRAM GB/s
PCIe data movement	PCIe bandwidth
Computational Depth	OMP Overhead
MPI Message Size	Network Bandwidth
MPI Send:Wait ratio	Network Gap
#MPI Wait's	Network Latency

LogP

Culler, et al, "LogP: a practical model of parallel computation", CACM, 1996.

Performance Model Retrospective

- Performance models often conceptualize the system as being dominated by one or more aspects of machine and application…

#FP operations	FLOP/s
Cache data movement	Cache GB/s
DRAM data movement	DRAM GB/s
PCIe data movement	PCIe bandwidth
Computational Depth	OMP Overhead
MPI Message Size	Network Bandwidth
MPI Send:Wait ratio	Network Gap
#MPI Wait's	Network Latency

LogGP

Alexandrov, et al, "LogGP: incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation", SPAA, 1995.

Performance Model Retrospective

- Architectural innovations → Throughput Limited Regime